Bevezetés

Az elmúlt években jelentos változásokon ment keresztül a mikroprocesszorok világa. Korunk információs forradalmának megfeleloen a személyi számítógépek ma már a háztartások nagy részében megtalálhatók. A mikroprocesszorok alacsony ára és megbízhatósága és az elektronikus szabályozások kedvezo tulajdonságai miatt szinte már minden háztartási eszközben megtalálhatóak. A hibaturo számítástechnika szintén jelentos fejlodésen ment át, aminek köszönhetoen már évek óta utazhatunk olyan repülogéppel, amelyben a mechanikus információ továbbítást elektronikus vezérlésre cserélték le. Az ilyen, ún. biztonság-kritikus rendszerekben (lásd később) a számítógépes rendszerrel szembeni meghibásodási követelmények megegyeznek a kiváltandó mechanikus rendszerek hasonló paramétereivel, továbbá nagy az igény a folyamatos rendelkezésre állással szemben is. Ezekkel a problémákkal a személyi számítógépek világában korábban nem szembesültek a tervezok, és új megoldásokra volt szükség, hogy megfeleljenek az új követelményeknek. A piac mérete is fejlesztésekre sarkallta a gyártókat. 2001-ben a világon 55,77 millió gépkocsit gyártottak. Ma már átlagosan minden legyártott autóban kb. 50 mikroprocesszor található. (Összehasonlításképpen: egy átlagos észak-amerikai család otthonában, ahol számítógép is megtalálható, szintén kb. 50 mikroprocesszor van) Észak-Amerikában minden évben kb. 3-szor annyi személy autót gyártanak, mint ahány házat építenek.

A beágyazott rendszerek legtöbbször zárt hurkú szabályozási köröket valósítanak meg. Biztonság-kritikus alkalmazások esetén mint például a ‘fly-by-wire’ vagy ‘drive-by-wire’ rendszerek esetén , ahol nincs közvetlen mechanikus kapcsolat a pilótafülke kapcsolószervei és a hajtómu ill. a fékpedál vagy gázpedál és a fék vagy a motor között, fontos szempont a nagy megbízhatóság és ezért a redundancia alkalmazása nehezen kerülheto el. A viszonylag egyszeru beágyazott rendszerbol így lesz elosztott rendszer, és a szabályozó rendszert ki kell egészíteni a redundancia intézés és hibaturés szükséges mechanizmusaival.

A megvalósított biztonság-kritikus rendszerekre jellemzo a hibaterjedés elleni fokozott védelem, a laza összeköttetésben lévo komponensek önmagukban hibaturo tulajdonságokkal rendelkeznek. Ezáltal a hibaterjedés lehetosége a minimálisra van szorítva, mivel nincsenek közös, megosztott eroforrások, ezért egy egység hibája kis hatással van a többi egység muködésére. A megoldás hátránya a magas ára, ezért a figyelem mostanában már a jobban integrált megoldások felé tolódott el, ami az eroforrások bizonyos szintu megosztásával jár együtt. Ezzel együtt viszont új hibaforrás jelenik meg, a hibaterjedés veszélye a rendszer alkotórészei között. Az integrált rendszerben a hibaterjedés elleni védekezést a rendszer megfelelo particionálásával lehet megoldani. A cél különálló egységek alkalmazása úgy, hogy futásuk közben az integrációból adódó egymásra hatásokat elhanyagolható ráfordítással valósítsák meg. A hibaturés, particionálás és integrálás problémái nagy kihívást jelentenek. Ha kezelésük ad-hoc módon történik, akkor ez a tény lesz a fo forrása a megvalósított rendszer hibáinak és megbízhatatlanságának. Szerencsére ezen problémák legtöbbje független magától a konkrét feladattól, és általános és elvileg helyes módon megoldható bizonyos mechanizmusok implementációjával. Egy tipikus példája az ilyen szolgáltatásoknak az információ továbbítása az elosztott rendszer egyik elemétol a másikhoz. A kommunikációs rendszer az egyik legfontosabb alapeleme a biztonság-kritikus elosztott rendszernek, fizikai médiától kezdve a kommunikációt biztosító protokollon keresztül. A szakirodalomban gyakran ezekre az architektúrákra mint buszrendszerekre hivatkoznak (ezzel is kiemelve a kommunikáció fontosságát), bár az elnevezés nem utal bonyolultságukra és kifinomultságukra. Valójában a buszok (protokollok) biztosítják a biztonság-kritikus jellemzoket az alkalmazások számára, és így a biztosított szolgáltatások választéka és implementációjuk mechanizmusa a biztonság-kritikus rendszerek értékelésének legfontosabb szempontjai között szerepelnek.

Alapfogalmak

Eloször érdemes a fontosabb fogalmakat tisztázni, mielott az idovezérelt protokollok ismertetésébe kezdenénk. A vizsgálandó rendszerek az *elosztott rendszerek*, azaz adott számítási teljesítménnyel rendelkezo entitások halmaza, melyek egy kommunikációs eroforrást felhasználva osztják meg egymás között az információt. A specifikáció teljesítése szempontjából mindegy, hogy a feladatot központosított vagy elosztott architektúrával valósítjuk meg. Azonban létezik egy architektúrális alapelv, mely szerint az objektum feladata megszabja fizikai megvalósítását is. Az elosztott rendszerek ennek a bizonyított alapelvnek az alkalmazását jelentik számítási feladatok egy részének megoldására.

Tegyük fel, hogy a feladat végrehajtására meghatározott ido áll rendelkezésre, van egy válaszido, amit a szolgáltatásnak teljesíteni kell. A feladattal szemben nem csak az a követelmény, hogy végrehajtsa meghatározott funkcionalitását, hanem hogy azt adott futási ido alatt tegye meg. Az ilyen rendszereket nevezzük *valós-ideju rendszereknek.*

Létezhetnek még elvárások a szolgáltatás minoségével szemben is. Például biztonság-kritikus rendszerekben ilyen elvárás a szolgáltatások idoszerusége és jósolhatósága. Azon valós-ideju rendszereket, amelyek legalább egy biztonság-kritikus feladattal rendelkeznek, *biztonság-kritikus* valós-ideju rendszereknek nevezzük. A biztonság-kritikus valós-ideju rendszereken belül további osztályozás teheto meg.

Ha a megadott válaszido nem teljesül, de a szolgáltatás még így is kiszolgálásra kerül (esetleg csökkentett módon) akkor gyengén valós-ideju (soft real-time) rendszerekrol beszélünk. Ha a rendszer nem képes tolerálni a válaszido nem teljesítésébol adódó problémákat, akkor szigorúan valós-ideju (hard real-time) rendszerekrol beszélünk. A biztonság-kritikus valós-ideju rendszerek többnyire szigorúan valós-idejuek.

Ha a rendszernek van egy biztonságos állapota, amelybe hiba észlelése esetén átvált, akkor a rendszer hiba-biztos (fail-safe). Ha még a hiba ellenére is muködoképes marad, akkor hiba-muködoképes (fail-operational).

Amikor biztonságos rendszerrol beszélünk, akkor a rendszer minden rétegének teljesítenie kell a szolgáltatások minoségével szemben támasztott követelményeket. Az alkalmazói réteg muködésének az idobeli elvárásoknak meg kell felelnie és jósolhatónak kell lennie. Például megfelelo idopontokban le kell kérdeznie az érzékelot, ez alapján helyes értékeket kell kiszámítania és végül megbízhatóan kell a beavatkozókat vezérelnie. A kommunikációs rétegnek az információ továbbítását kell megfelelo idoben végrehajtania.
Ha valamilyen szolgáltatást várunk el még abban az esetben is, ha hiba történt a rendszerben, akkor a rendszer tervezés során egy *hiba hipotézist* használunk, ami megadja, hogy milyen hibákat kell túlélnie a rendszernek. Egy gyakran használt hiba hipotézis az egyszeres hiba hipotézis: bármilyen egyszeres hiba elofordulhat a rendszerben, de két független hiba nem fog elofordulni a rendszerben egy bizonyos idon belül. Ez az ido pedig elég a rendszer muködoképes, normális állapotába történo visszatéréséhez. Ha egy biztonság-kritikus valós-ideju rendszer nem tud hiba esetén biztonságos állapotba átváltani, akkor túl kell élnie minden egyszeres hibát. Az egyszeres hibát toleráló rendszereknek *redundanciát* kell alkalmazniuk a muködoképesség megorzésére. A redundancia nem növeli a rendszer funkcionalitását, csak a rendelkezésre állását. Az architektúra szempontjából a duplikált vagy egyéb módon redundáns csomópontokat, amelyek egy feladatot látnak el, hibaturo egységeknek (Fault-Tolerant Unit, FTU) nevezzük.
Feltételezve, hogy egy csomópont hibátlanul muködik, hiba esetén különbözo hibás állapotokba kerülhet:

- *hiba-kontrollálatlan* (fail-uncontrolled): a rendszer fennmaradó része eltéro véletlenszeru állapotokat érzékel a hibás csomópont állapotáról (az ilyen hibákat nevezik Bizánci típusúaknak). Legalább 4 csomópontból álló FTU szükséges egy hiba-kontrollátlan állapot tolerálására.
- *hiba-konzisztens* (fail-consistent): a rendszer többi része konzisztens információt kap a hibás csomópontról, de nem feltétlenül érzékeli, hogy meghibásodott a csomópont. Háromszoros redundancia használatával az FTU tolerálja ezt a hibát.
- *hiba-korlátozott* (fail-restrained): a rendszer hibátlanül muködo csomópontjai egy ideig eltéro információkkal rendelkeznek a hibás csomópontról, de egy idokorláton belül konzisztens nézet alakul ki a meghibásodott csomópontról. Az idokorlát elmúlásával egy duplikált csomópontot tartalmazó FTU képes a hibát elviselni.
- *hiba-csendes* (fail-silent): a rendszer konzisztens információt kap a csomópont meghibásodásáról. Szintén kétszeresen redundáns csomópont segítségével elviselheto a hiba.

A késobbiekben ismertetésre kerülo TTP/C protokoll a következo hiba turési képességekkel rendelkezik:

- hiba-csendes az idotartományban
- hiba-konzisztens az alkalmazói értéktartományban
- egy-csatornás Bizánci hibák teljes elviselése
- két-csatornás hibák hiba-korlátozott elviselése

Köszönhetoen az alkalmazói értéktartományban biztosított hiba-konzisztenciának, a szavazás biztonságosan elvégezheto a vevo oldalán (mindenhol ugyanaz az állapot látott a rendszerben). Az alkalmazói réteg megfelelo tervezésével még az is biztosítható, hogy a hiba-konzisztencia viselkedés az értéktartományban hiba-csendessé növelheto.

A kommunikáció idovezérelt megközelítése

A valós-ideju rendszert idovezéreltnek nevezzük, ha a vezérlo jelek a rendszerszintu globális ido múlása miatt következnek be, és olyan folyamatokat vezérelnek, mint például üzenetek küldése és vétele, taszkok aktivációja, illetve külso állapotváltozások észlelése lekérdezéssel vagy mintavételezéssel. Az események pontos idopontja, bekövetkezése idovezérelt rendszerben nem lényeges. Esemény-vezérelt valós-ideju rendszerekben a vezérlo jelek az események bekövetkezésébol adódnak, mint például üzenet fogadása, külso megszakítás által jelzett állapot változás, helyi idozíto jelzése. A nem valós-ideju rendszerek mindig esemény-vezéreltek.
Az idovezérelt rendszerek további tárgyalásához az állapot és az esemény fogalmát kell tisztázni. A rendszer elméletben az állapot szolgál a múlt és a jövo szétválasztására: az állapot lehetové teszi a jövo meghatározását a jelenlegi állapot és a jövobeli bemenetek ismeretében. Az állapot egy olyan körülmény, ami fennáll egy ido intervallum erejéig. Az esemény egy fontos, jelentos történés az idotengely mentén.
Egy interfészen keresztül továbbított információ lehet állapot információ, vagy pedig esemény információ. Az állapot információ egy állapot jellemzoit tartalmazza a megfigyelt idopontban, abszolút értékben. Például az, hogy a szobában 20 ?C van, egy állapot információ. Az esemény információ ezzel szemben az állapotban bekövetkezett változásról szolgáltat információt, azaz egy relatív jellemzo. Az elozo példa analógiájára az a tény, hogy a homérséklet +1 ?C-al megváltozott, egy esemény információ. Az állapot üzenetek az idovezérelt kommunikációra jellemzoek, mert állapot információkat tartalmaznak, és a küldo által periodikusan frissítheto még akkor is, ha nem történt állapot változás. Ezzel szemben az esemény üzenetek gyakran szerepelnek esemény-vezérelt kommunikációban, mert esemény információt tartalmaznak, tipikusan aperiodikusan kerülnek küldésre, és küldo sor illetve vételi sor szükséges a küldéshez, fogadáshoz. Valós-ideju alkalmazásokban a tervezo megközelítésétol függoen mind az állapot mind pedig az esemény információ használható. Ha az esemény-vezérelt üzenetek összeütköznek az idovezérelt üzenetekkel, akkor a rendszer viselkedése függ az üzenet átvitel idobeli jellemzoitol. Az ilyen rendszerek hibaturo képességeit nagyon nehéz megjósolni vagy validálni.

Az idovezérelt architektúra (Time-Triggered Architecture, TTA) koncepciója a következokbol áll:

- a kommunikációs rendszer idovezérelt módon muködik, rendelkezésre áll egy globális ido-alap, és minden kommunikációs aktivitás erre a globális idore alapul
- az alkalmazói rendszer felhasználja ezt a kommunikációs rendszert, és saját feladatait (taszk aktiváció, stb.) a globális idore alapozza
- az alkalmazás és a kommunikációs rendszer között egy alaposan definiált, tervezési idoben eldöntött jellemzokkel bíró interfész helyezkedik el, ami az alkalmazás és a kommunikációs rendszer számára egyaránt ismert

A következo táblázat végül összefoglalja az idovezérelt és az esemény-vezérelt kommunikáció fontosabb jellemzoit.

Idovezérelt kommunikáció	**Eseményvezérelt kommunikáció**
meghatározott üzenet látenciák	terhelés függo üzenet látenciák
konstans eroforrás felhasználás	változó eroforrás felhasználás
ismert csúcs-és túlterhelési viselkedés	nehéz a csúcs- és túlterhelés szabályozása
nagyfokú jósolhatóság	alacsonyfokú jósolhatóság
egyszeru az idobeliség ellenorzése	sok teszt állapot
bovítés csak akkor lehetséges, ha elore tervezett	könnyu bovíthetoség
kis rugalmasság	nagy rugalmasság

Az esemény-vezérelt rendszerek az eroforrások rugalmas kihasználását nyújtják, ami nagyon vonzó, ha a követelmények változóak. Azonban biztonság-kritikus alkalmazásoknál fontos bizonyos minimális szolgáltatási minoség biztosítása a rendszer komponenseinek, még hiba esetén is. A busz elrendezés csomópontjai valós-ideju beágyazott rendszerek, ezért a minimálisan elvárt szolgáltatás a megjósolható kommunikáció alacsony látenciával és jitter-rel. Az esemény-vezérelt buszok fo problémája, hogy a különbözo csomópontokon bekövetkezo események miatt a buszhoz egyszerre többen szeretnének hozzáférni, ezért valamilyen média hozzáférés szabályozásra van szükség, hogy minden csomópont megszakítás nélkül képes legyen a buszt használni. Lényeges kérdés, hogy a csomópont szemszögébol a busz hozzáférés megjósolható-e, és hogy hiba esetén is fennállnak-e az elorejelzések. A buszok egy része, mint például az Ethernet, egy valószínuségben állapítják meg az idobeli hozzáférés garanciáját, és hiba esetén meg semmilyen biztosítékot nem nyújtanak. A nem biztonság-kritikus beágyazott rendszerek, mint például a CAN, LonWorks vagy Profibus, prioritásokkal, elore meghatározott idorésekkel vagy zsetonokkal biztosítják a hozzáférés determinisztikusságát. Például a CAN esetében a legnagyobb prioritású üzenet mindig gyoz, így csak a folyamatban lévo adatátvitel végét kell megvárnia. Azonban a látencia terhelésfüggo lesz, és csak valószínuségi korlátok állíthatók. Az idovezérelt rendszerek statikus kommunikációs sávszélesség allokációt alkalmaznak globális ütemezés révén, minden csomópont ismeri az ütemezést és ismeri az idot, így pontosan tudja, hogy mikor küldhet üzenetet, és mikor számíthat üzenetek vételére. Ezért a hozzáférés arbitráció tervezési idoben statikusan eldol, és így a következmények elore megvizsgálhatók. A statikus ütemezés a fecsegési (bubbling idiot) hiba kezelését is lehetové teszi. Ez a busz orzok alkalmazásával valósítható meg, ami csak akkor engedélyezi a busz hozzáférést, amikor az az ütemezésben megengedett. A busz orzonek tudnia kell, hogy mikor férhet hozzá a csomópontja a buszhoz, amit esemény-vezérelt esetben nagyon nehéz meghatározni, és koncepcionálisan egyszeru az idovezérelt esetben.
A kommunikációt maga az ido és az erre alapuló statikus ütemezés vezérli, így nincs szükség speciális azonosítókat csatolni az üzenethez, mint például küldo és címzett, mert a küldés ideje mindezen jellemzoket meghatározza. Ez nem csak egyszeruen a sávszélesség növekedéséhez vezet, hanem egy nagyon fontos hibaforrást is megszüntet: annak a lehetoségét, hogy egy hibás csomópont másnak küldjön üzenetet, illetve hogy egy hibás csomópont más csomópontot személyesítsen meg.

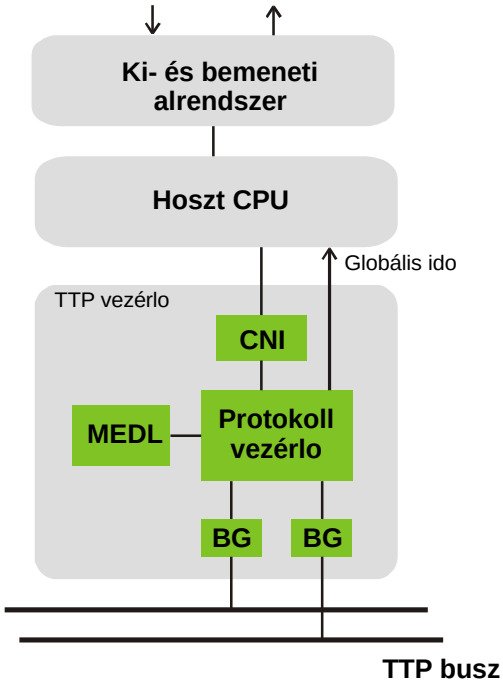
Az idovezérelt protokoll

Az idovezérelt protokollok (Time-Triggered Protocols) új megközelítését jelentik a hibaturo valós-ideju elosztott rendszerek által támasztott speciális követelményeknek. Az idovezérelt protokolloknak több megvalósítása is létezik, ezekben azonban sok a közös vonás. Itt most a TTP/C protokollnak megfelelo elnevezések segítségével ismertetjük az idovezérelt protokoll vázát, beleértve a TTP/C protokoll sajátosságainak ismertetésével, amelyik az egyik legjobban dokumentált és legegyszerubben elérheto rendszer.
A szokásos OSI 7 rétegu modelltol eltér a TTP egy csomópontjának felépítése, az egyetlen hasonlóság a fizikai és adat réteg jelenléte mindkét modellben. A TTP a következo rétegekbol áll: SRU réteg, RM réteg, FTU réteg. A legkisebb helyettesítheto egység (Smallest Replaceable Unit - SRU) réteg felelos a konzisztens tagsági függvény fenntartásán. A redundancia intezo (Redundancy Managment - RM) réteg biztosítja a rendszer újrakonfigurálását és a rendszer indításakor szükséges adminisztratív feladatokat látja el. A hibaturo egység (Fault-Tolerant Unit) réteg képes több csomópontot egy megnövelt hibaturésu egységbe szervezni. A TTP rendszer egy csomópontjának a legfontosabb része

a kommunikációs-hálózati illeszto (Communication Network Interface). A CNI szerepe elvi jelentoségu, a kommunikációs-hálózati illeszto egy idoablakot biztosít a hoszt számítógép részére, amelyben hozzáférhet a kommunikációs csatornához.

Az idovezérelt rendszer struktúrája

Az idovezérelt rendszer egy elosztott valós-ideju rendszer. Ennek megfeleloen egy klaszter csomópontokból áll, melyek egy kommunikációs hálózaton keresztül kapcsolódnak össze. A csomópontok önálló számítási teljesítménnyel rendelkezo eroforrások, a kommunikációs rendszer pedig a TTP busz. A kommunikációs rendszer topológiája lehet busz, vagy csillag. A csomópont maga a legkisebb helyettesítheto egység (SRU), mely meghibásodás esetén kicserélheto, vagy átkonfigurálható. A csomópont két alrendszert tartalmaz, a hoszt számítógépet, és a kommunikációs vezérlot. A kommunikációs-hálózati vezérlo a csomóponton belüli összeköttetést biztosítja a két alrendszer között. Megvalósítása legtöbbször egyszeruen egy duál-portos memória egység (DPRAM), amely mindkét alrendszer által írható. Természetesen a CNI esetében biztosítani kell a kölcsönös kizárást, illetve az adatok szinkronizációját. Az adatok integritását a hoszt és a kommunikációs vezérlo között egy speciális algoritmussal, a nem-gátolt írás (NBW) prokollollal biztosítják.

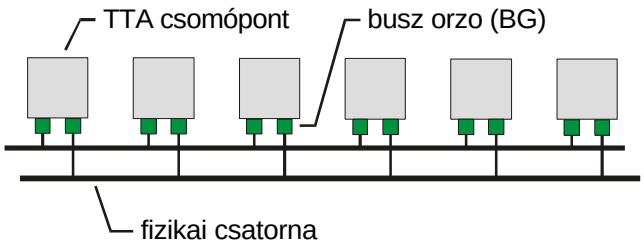


Ábra 1. Csomópont felépítése

A kommunikációs vezérlo része az üzenet-leíró tábla (Message Desciptor List - MEDL). Ez a táblázat határozza meg, hogy egy csomópont mikor küldhet el egy üzenetet, és mikor várhat üzenetet egy másik csomóponttól. A MEDL mérete egy klaszter ciklus, ami TDMA körök sorozatából áll. Ezen kívül a kommunikációs vezérlo egység tartalmaz még csatornánkét egy busz orzo-t (Bus Guardian - BG), melyek ellenorzik az idobeli hozzáférést a csatornákhoz, és felfüggesztik a vezérlo muködését helytelen hozzáférés esetén.
A TTP egy idoosztásos-többszörös hozzáféréses (Time-Division Multiple Access - TDMA) protokoll, ahol minden csomópont küldhet üzenetet a megosztott kommunikációs csatornán egy elore rögzített statikus idobeosztásnak megfeleloen.

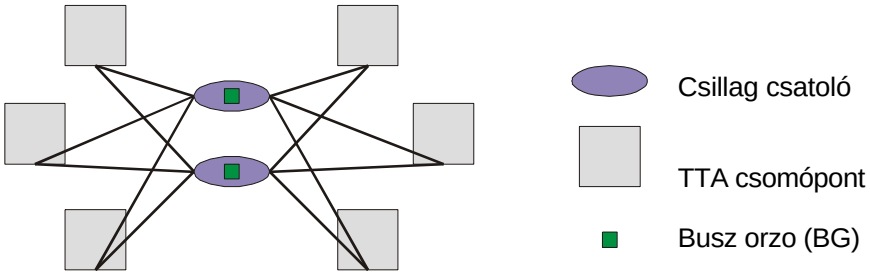
Topológia

Az idovezérelt architektúra egy klaszterében két összeköttetési topológia között dönthetünk: a TTA-busz vagy a TTA-csillag.
A TTA-busz struktúrában a fizikai összeköttetés a csomópontok között replikált passzív buszokon keresztül valósul meg. Minden fizikai csomóponton legalább 3 alrendszer található meg: két busz orzo és maga a csomópont. A busz orzok olyan független egységek, amelyek felügyelik a csomópont a priori ismert temporális viselkedését. Ha egy csomópont üzenetet kíván küldeni az elore meghatározott idoablakán kívül, akkor a busz orzo megtiltja a fizikai csatornához történo hozzáférést, és megszünteti ezt a hibát a klaszter szempontjából. Ideális esetben a busz orzoknek teljesen független egységeknek kell lenniük önálló oszcillátorral, tápegységgel és óra szinkronizációs algoritmussal. Továbbá, a busz orzoknek olyan fizikai távolságban kell lenniük a csomóponttól, hogy ellenállók legyenek a térbeli közelségbol adódó hibákra. Ha mindezek a követelmények megvalósulnak egy TTA csomóponton, akkor 3 független integrált áramkört, 3 független órát és 3 független tápegységet kell a csomópontnak tartalmaznia. Ez természetesen jelentosen megdrágítja a megvalósítást. Azért, hogy az implementáció költségeit csökkentsék, a busz orzoket is a kommunikációs vezérlo lapkáján valósítják meg.



Ábra 2. TTA-busz topológia

A TTA-csillag topológia is replikált adatátviteli csatornákat használ, amelyek egy csillag csatolón keresztül kapcsolódnak össze. Ebben az esetben a busz orzok a csillag csatolókon helyezkednek el. A csillag kapcsolás ezért néhány el nem hanyagolható elonnyel rendelkezik a busz topológiával szemben. Csillag kapcsolás esetén csak két busz orzore van szükség, függetlenül a csomópontok számától. A busz orzok valóban teljesen függetlenek a csomópontoktól, és fizikailag is megfelelo távolságban vannak a védett csomópontoktól. A busz orzok további diagnosztikai feladatokat is elláthatnak feladataik kiterjesztésével. Ha a busz orzok a fizikai jeleket helyreállítják, akkor bizonyos hibákkal szemben rugalmassá válik az architektúra. A pont-pont összeköttetésnek jobbak az elektro-magnetikus interferencia jellemzoi, és könnyedén implementálható üvegszálas fizikai csatornával.



Ábra 3. TTA-csillag topológia

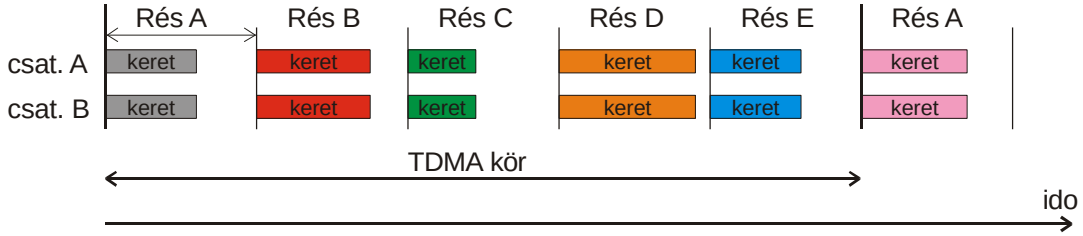
Prokoll változatok

Két változata alakult ki az idovezérelt protokolloknak: a teljes funkcionalitást biztosító TTP/C, és az egyszerubb TTP/A. A kommunikációs-hálózati illeszto struktúrája viszont kompatibilis mind a két esetben.

- A TTP/C segítségével megvalósítható egy hibaturo elosztott valós-ideju rendszer. A TTP/C támogat többféle replikációs stratégiát. Az ár, amit ezért fizetni kell, egy dedikált kommunikációs vezérlo használata, amely implementálja a protokoll nyújtotta szolgáltatásokat.
- A TTP/A egy nem hiba-turo terepi busz, mely megvalósításához mindösze egy szabványos UART vezérlo, egy lokális valós-ideju óra szükséges. A korszeru mikronkontrolleken ezek az egységek mind megtalálhatók. Maga a protokoll pedig egy szoftver, ami a mikrokontrolleren fut.

Adatátvitel a TTP/C protokollban

A TTP/C protokoll idoosztásos többszörös hozzáféréses (TDMA) protokoll-t használ az üzenetek továbbítására. Az adatátvitel az idotartományban ennek megfeleloen feloszthatóa következo egységekre: klaszter ciklus, TDMA kör és idorés. Minden csomópontnak ki van jelölve egy idorés. Elofordulhat olyan eset is, hogy több csomópont osztozik egy idorésen, ezért az idorés nem feltétlenül azonosítja magát a csomópontot is. De minden csomópont a klaszter ciklus során legalább egyszer hozzáférhet elore meghatározott módon egy idoréshez, azaz részt vehet a kommunikációban. A TTP/C kommunikációs alap ciklus ideje a TDMA kör. A TDMA kör során minden csomópont legfeljebb egyszer hozzáférhet a kommunikációs közeghez, azaz a TDMA kör az idorések egy sorozata. Ha minden csomópontnak különbözo idorés van kijelölve, akkor a TDMA kör annyi idorésbol áll, ahány csomópont van a rendszerben.
A hozzárendelés statikus, a rendszer tervezésekor eldol, hogy melyik csomópont mikor használhatja az adatátviteli csatornát, és ezen az ütemezésen késobb már nem lehet változtatni. Viszont ezáltal minden csomópont számára egy garantált sávszélesség áll rendelkezésre az információ továbbítására. Nincs megkötés az idorések nagyságára vonatkozóan, lehetnek csomópontonként különbözoek is, viszont nagyságuk statikusan meghatározott. Az idorés ütemezés a minden csomóponton megtalálható kommunikációs vezérlo üzenetleíró táblázatában (MEDL) kerül tárolásra. Általában minden statikus ismeret az idovezérelt kommunikációval kapcsolatban a MEDL-ben tárolódik.

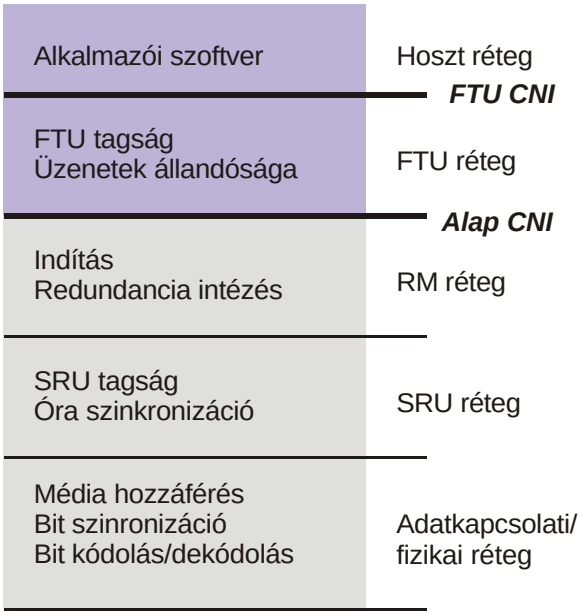


Ábra 4. Adatátviteli séma

Minden résben egy-egy keret küldése történik meg két csatornán. A keretek küldése szigorúan az idorés legelején történik meg. A keretek küldése között mindig van egy ido intervallum, amíg nincs adatátvitel a csatornán. Ez az ido a keret-közi ur (inter-frame gap), amely a kommunikációs vezérlo számára szükséges, hogy bizonyos számításokat elvégezzen.
A klaszter ciklus egymás utáni TDMA körök sorozata. A TDMA körökben továbbított üzenetek eltéroek is lehetnek, nincs megszabva, hogy minden idorésben minden TDMA körben ugyanazt az üzenetet kell továbbítani. A klaszter ciklus viszont már periodikusan ismétlodik az adatátvitel során, ezért a klaszter ciklus során minden továbbítandó üzenet legalább egyszer elküldésre kerül. Ez meg is szabja az üzenetküldés frekvencia tartományát: leggyakrabban minden TDMA körben továbbítható egy üzenet, illetve a legritkább esetben minden klaszter ciklus alatt egyszer. Az alkalmazói oldalról még tovább csökkentheto a küldési frekvencia (például multiplexelt üzenetek révén), azonban a maximális küldési frekvencia kötött: a TDMA körnél gyorsabb adatátvitel nem képzelheto el.

A TTP/C rétegei

Az OSI modellhez hasonlóan a TTP/C kommunikáció is rétegekbe szervezheto.



Ábra 5. TTP/C rétegei

Adatkapcsolati/fizikai réteg

Az adatkapcsolati réteg végzi a keretek továbbítását a csomópontok számára. Fontosabb feladatai a fizikai médiához történo hozzáférés biztosítása, bit szinkronizáció biztosítása és bit kódolás/dekódolás végrehajtása. A csatorna hozzáférés típusa idoosztásos többszörös hozzáférés (TDMA), ahol a hozzáférés a kommunikációs vezérloben található üzenet-leíró tábla alapján történik. A bit szinkronizáció és bit kódolás/dekódolás alapja a módosított frekvencia moduláció kódolás (Modified Frequency Modulation). Csavart érpáros megvalósítás esetén a fizikai réteg akár a CAN busz fizikai rétegével is megegyezhet, mivel a TTP rendszer gyengébb követelményeket támaszt a fizikai réteggel szemben, mint amit a CAN protokoll fizikai rétege nyújt.

Legkisebb helyettesítheto egység (SRU) réteg

A vett keretekbol az SRU réteg végzi el az üzenetek kibontását az üzenet-leíró táblában található információ alapján, és az üzeneteket elhelyezi a kommunikációs-hálózati illesztoben (CNI). Szintén az SRU réteg tartja karban a csomópont tagságot. Az óra szinkronizációt biztosító hibaturo algoritmus is a réteg részét képezi. A módváltásokhoz szükséges

szolgáltatásokat is nyújt a felsobb rétegek számára. Az azonnali módváltások rögtön kiszolgálásra kerülnek, míg a késleltett módváltások a következo klaszter ciklus kezdetéig fel vannak függesztve.

Redundancia intézo (RM) réteg

A redundancia intézo réteg hajtja végre a TTP/C rendszer hideg indítását. Az RM réteg használja fel az SRU réteg által biztosított módváltás szolgáltatást. A megjavult csomópont újraintegrálását is a redundancia intézo réteg végzi el. A dinamikus redundancia kiszolgálással kapcsolatos teendoket is az RM réteg látja el, például egy meghibásodott csomópont helyettesítését az árnyék csomópontjával. Erre a célra a csomópont újrakonfigurálás mezo-t biztosítja a CNI. Amennyiben a hoszt úgy dönt, hogy egy új csomópont feladatra van szükséges, akkor a kívánt feladat azonosítóját a hoszt beírja ebbe a mezobe. A kommunikációs vezérlo ellenorzi, hogy engedélyezheto-e az új csomópont feladat. Ha igen, akkor végrehajtja a csomópont feladat váltást, majd felprogramozza a busz orzoket a busz hozzáférés ellenérzésére az új helyzetben.
Tipikus szerepe az RM rétegnek a meghibásodott aktív csomópont helyettesítése az árnyékcsomópontjával (természetesen amennyiben van árnyékcsomópontja). Az árnyékcsomópont RM rétege észleli az aktív csomópont meghibásodását, és újrakonfigurálást kezdeményez, hogy átvegye a meghibásodott csomópont feladatát. Újrakonfigurálás után a rendszer újra muködoképes.

Hibaturo egység (FTU) réteg

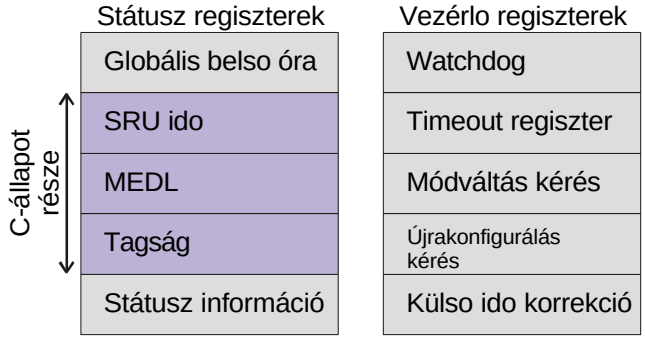
A hibaturo egység réteg képes több csomópontot egy megnövelt hibaturésu muködési egységbe szervezni. Az FTU réteg biztosítja, hogy az FTU CNI-ban megtalálható adatok mindig konzisztensek legyenek, azaz csak akkor íródnak át az alap CNI-ból, ha már minden üzenet megérkezett a csomóponthoz (vagy már biztos, hogy sosem fog megérkezni). Például 3 csomópont segítségével kialakítható a háromszoros moduláris redundancia (Triple Modular Redundancy) hibaturo egység, ahol a 3 egység szavazása alapján dol el, hogy mi is a vett üzenet. A TMR FTU egység képes egyszeres hibák tolerálására. Egyszerubb esetekben az is megoldható, hogy a csomópontokon futó bizonyos alrendszerek több csomóponton párhuzamosan fussanak, amely adminisztrációját szintén az FTU réteg végzi el.
Az FTU tagság az FTU réteg szolgáltatása. Az FTU réteg implementációja elképzelheto mind a hoszt processzoron, mind pedig a kommunikációs vezérloben. A jelenleg kapható TTP/C vezérlokben nincs implementálva az FTU réteg, csak az alap CNI-t biztosítja a hoszt számítógép számára.

Az alap CNI

A kommunikácós-hálózati illeszto a legfontosabb része az idovezérelt architektúrának, mivel az alkalmazói szoftver számára mindössze ez a rész látható a kommunikációs hálózatból.
Az alap CNI biztosítja az információ cserét az RM réteg és az FTU réteg között. Két fo része van: a státusz és vezérlo regisztereket tartalmazó Státusz/Vezérlo Terület, amely a rendszerre vonatkozó információkat tartalmazza, és az Üzenet Terület, amely tartalmazza a vett és küldött üzeneteket.
Az alap CNI interfészen kívül mindössze egyetlen vezérlo jel teremt kapcsolatot a kommunikációs vezérlo egység és a hoszt processzor között, ami a globális idot adja át az hoszt számítógépnek.

Státusz/Vezérlo terület

A státusz/vezérlo terület tartalmazza az idovezérelt rendszerre vonatkozó információkat, illetve ezen keresztül kezdeményezheto módváltás vagy újrakonfigurálás.



Ábra 6. Az alap CNI státusz és vezérlo regiszterei

A státusz regisztereket a kommunikációs vezérlo frissíti.A globális belso óra tartalmazza a 2 byte-os globális rendszer idot, mely a rendszerben található kommunikációs vezérlok kölcsönös szinkronizációja alapján lesz meghatározva. A következo 3 regiszter tartalmazza a vezérlo állapotát (controller state, C-state, C-állapot), mely regisztereknek kitüntetett szerepük van. A három regiszter az SRU ido regiszter, az üzenetleíró tábla (MEDL) mutató, és a csomópont tagság vektor. Az SRU ido tartalmazza a globális idot SRU idoablak felbontásban. Érteke konstans egy SRU idoablak alatt, és a következo SRU idoablak kezdetekor kerül növelésre. A MEDL pozíció adja meg az aktuális rendszer üzemmódot, és azt, hogy mi az aktuális pozíció a MEDL-ben. A csomópont tagság vektorban minden csomópontnak

megfelel egy bit, és ennek megfeleloen annyi bitból áll, ahány csomópont van a rendszerben. Amennyiben egy csomópontnak megfelelo bit értéke ‘1’, akkor a csomópont az utolsó idorésében jól muködött (pl. üzenetet küldött). Ellenkezo esetben (bit értéke ‘0’) az utolsó idorésében már nem muködött hibátlanul. A tagsági vektor frissítése minden idorés végén történik meg, amikor már minden küldött üzenetnek meg kellett érkeznie, és az üzenetek ciklikus redundancia kódjai ismertek. A protokoll csak akkor muködik hibátlanul, ha a rendszerben konzisztens a C-állapot vektor.
A vezérlo regisztereken keresztül a hoszt képes vezérlo információt átadni a kommunikációs vezérlonek. A watchdog mezo-t a hosztnak folyamatosan frissítenie kell, amit a TTP/C vezérlo folyamatosan ellenoriz. Így gyozodik meg a vezérlo arról, hogy a hoszt muködik-e egyáltalán vagy nem. Ha nem frissul periodikusan a watchdog mezo, akkor a kommunikációs vezérlo felfüggeszti az üzenetek küldését, és azt feltételezi, hogy a hoszt számítógép meghibásodott. A timeout regiszter felhasználásával kérhet a hoszt megszakítást egy meghatározott jövobeli idopontban. Ezáltal lehetové válik a hoszt muködését a globális idohöz hangolni. A hoszt processzor beírja ebbe a regiszterbe a kívánt megszakítás idejét. Amikor a globális ido és a timeout regisztert értéke megegyezik, akkor a TTP/C vezérlo megszakítást kér. A módváltás regiszter alkalmazható új rendszer-ütemezés igénylésére. A módváltás igény ténye az igénylo csomópont idorésében kerül továbbításra a többi csomópontnak. Kétféle módváltás különböztetheto meg: azonnali, és késleltetett. Az azonnali módváltás azonnal kiszolgálásra kerül a többi csomópont által. A késleltetett váltás csak a következo klaszter ciklus kezdetekor kerül kiszolgálásra. Az újrakonfigurálás igény mezoben jelezheti a hoszt, hogy új feladatot kell kijelölni a csomópontnak. Például ha egy csomópont észreveszi, hogy egy kritikus csomópont meghibásodott, akkor a csomópont kérheti a kritikus, de meghibásodott csomópont feladatainak átvételét. A külso ido korrekció mezore az óra szinkronizáció miatt van szükség.

Üzenet mezok

Az üzenet mezok felépítése alkalmazás függo, hiszen a rendszerben továbbított üzenetek jellemzoi a tervezo által definiáltak. Az üzenetmezok struktúráját az üzenetleíró tábla határozza meg. Az üzenet mellett még egy státusz byte is tárolódik, ami potenciális hiba eseményekrol tájékoztat.

Az adatátvitel konzisztenciája (a hoszt és a TTP/C vezérlo között)

A TTP/C vezérlotol a hoszt felé irányuló adatátvitelt az üzenetleíró táblázat határozza meg. A vezérlo belso vevo tárolóegységébol az üzenet egy elore meghatározott idopontban íródik át a CNI üzenet mezojébe. Az átírással együtt a státusz byte értékét is frissíti a vezérlo. Ez a folyamat minden SRU idorés vége elott következik be.
Mivel a hoszt processzor számára is ismert ez az a priori idopont, amikor az átírás megtörténik, ezért a hozzáférési problémák elkerülhetok ennek figyelembe vételével. Ha a hoszt processzor véletlenszeruen kíván hozzáférni a CNI-hoz, akkor a nem-gátolt írási (non-blocking write, NBW) protokoll-t kell használnia. Ezáltal észlelheto a TTP/C vezérlo bármilyen írási folyamata miközben a hoszt processzor olvassa a kommunikációs-hálózati illesztot.
A hoszt processzor szintén a priori ismeri, hogy mikor fogja a kommunikációs vezérlo a CNI-t olvasni. A hoszt operációs rendszer feladata azt biztosítani, hogy amíg a vezérlo olvassa a kommunikációs-hálózati illesztot, addig a hoszt ne írja a CNI-t.
Ha a hoszton futó szoftver idovezérelt muködésu, és a kommunikációs rendszer globális órájával szinkronban van, akkor a hoszton futó taszkok megfelelo ütemezésével elore biztosítható, hogy ne legyen olvasási/írási konfliktus a CNI-ban. Amennyiben a hoszt szoftver esemény-vezérelt, akkor egy retesz-mentes szinkronizációs (lock-free synchronization) protokollt kell alkalmazni, ami sohasem gátolja az írót (például magát a kommunikációs rendszert). Vizsgáljuk meg a nem-gátolt írási protokoll muködését abban az esetben, ha adatátvitel történik a kommunikációs-hálózati illeszton keresztül a kommunikációs rendszertol a hosz felé. Ebben az esetben egy író van, a kommunikációs rendszer, és sok olvasó, a hoszton futó taszkok sokasága. Az olvasó nem teszi tönkre az író által beírt információt, de az író megzavarhatja az olvasási folyamatot. A nem-gátolt írási protokollban az írás nincs akadályozva. Amikor egy új üzenet érkezik, akkor a kommunikációs rendszer rögtön beírja azt a kommunikációs-hálózati illesztot jelento duál-portos memória egységbe. Ha az olvasó az írási folyamat közben próbál meg egy üzenetet kiolvasni, akkor inkonzisztens információt kaphat, amit el kell dobni. Ha az olvasó képes az írást detektálni, akkor az olvasó újra kezdheti az olvasást mindaddig, amíg konzisztens információhoz nem jut.
A protokoll megvalósításához szükség van egy státusz mezore, a konkurencia ellenorzo mezore (concurrency control field, CCF), minden üzenethez. A mezohöz történo atomikus hozzáférést a hardvernek kell garantálni. A CCF kezdeti értéke nulla. Az írási folyamat kezdete elott az író megnöveli a mezo értékét, majd ha befejezte az írást, akkor újra megnöveli az értékét. Az olvasó a CCF értékének olvasásával kezdi az olvasási folyamatot. Ha a CCF értéke páratlan, akkor rögtön újra kezdi az olvasást, mert éppen írás van folyamatban. Az olvasás végén a CCF értékét újra ellenorizni kell. Ha idoközben megváltozott, akkor ismét újra kell kezdeni az egész olvasást, amíg nem sikerül egy zavartalan olvasást végrehajtani. Megmutatható, hogy létezik egy felso határa az olvasási próbálkozások számának, ha a két írási folyamat közt eltelt ido jelentosen nagyobb, mint maga az írási vagy olvasási folyamat. A nem-gátolt szinkronizációt használó eljárásokkal általában az a tapasztalat, hogy többnyire elonyösebbek, mint a gátló szinkronizációt használó eljárások (például multimédia rendszerek).

A TTP/C belso muködése

Üzenet leíró táblázat (MEDL)

Az üzenet leíró táblázat egy statikus adat struktúra, ami szabályozza, hogy mikor kell egy üzenetet küldeni vagy venni a kommunikációs csatornán keresztül. Hosszát a klaszter ciklus ido határozza meg, azaz azon TDMA körök sorozata, melyek ismétlodnek a rendszer muködése alatt. Az üzenet leíró táblázat egy bejegyzése 3 mezobol áll: egy ido mezobol, egy cím mezobol és egy attributum mezobol. Az ido mezo tartalmazza SRU ido felbontásban azt az idopontot, amikor a cím mezo által meghatározott üzenetet venni/továbbítani kell. A cím mezo mutat a kommunikációs-hálózati illeszto azon memória területére, ahol az üzenet megtalálható. Végül az attributum mezo az üzenet 4 jellemzojét tartalmazza. Az irány almezo határozza meg, hogy vett vagy küldött üzenetrol van-e szó. A hossz almezo adja meg az üzenet hosszát, az inicializálás almezo dönti el, hogy üzenet típusa inicializáló-e vagy nem. Végül van egy további paraméter almezo, ami mód váltáskor és feladat váltáskor használható fel. A hoszt csak olyan mód váltásokat juttathat érvényre, amelyek ezen almezo által engedélyezettek.
Az üzenet leíró táblázat fizikai megvalósítása függ a felhasznált kommunikációs vezérlotol. Minden csomópontnak egyedi üzenet leíró táblázata van a rendszerben.

Keret formátum

Hibátlan muködés esetén egy csomópont mindkét fizikai csatornán egy-egy keretet továbbít. Egy TTP/C keret tartalmaz egy fejlécet, magát az adatokat (üzenetek), és egy két (esetleg három) byte-os CRC kódot. A fejléc elso bitje dönti el, hogy inicializáló (I-keret) vagy normál (N-keret) keretrol van-e szó. A következo 3 bit használható fel mód váltás kérésére. Összesen 7 különbözo mód képzelheto el. A mód váltás engedélyezheto vagy tiltható az üzenet leíró táblázatban.

CRC számítás

Az inicializáló keretek számítása a fejléc és az adat byte-ok alapján történik. Normál keretek esetén a CRC számítása nemcsak a fejléc és az adat byte-ok alapján történik. A klaszter hibátlan muködése esetén a vezérlo állapotainak meg kell egyezniük (C-állapot), és errol meg is kell gyozodni. Elvileg minden keretnek tartalmaznia kellene a C-állapotot, és így könnyedén ellenorizheto lenne a C-állapot konzisztenciája a rendszerben. A konzisztencia ellenorzés azonban megoldható a C-állapotok küldése nélkül is. A küldo csomóponton a normál keret CRC kódja az adat byte-okat és a küldo C-állapotát tartalmazó vektoron számított. A vevo csomóponton a CRC kód a vett adat byte-ok és a vevo C-állapota alapján kerül kiszámításra. Ha a CRC ellenorzés negatív, akkor vagy a C-állapot nem konzisztens a rendszerben, vagy pedig a küldött adat sérült meg az átvitel alatt. Mindkét esetben a vett adatokat el kell dobni.

Tagság számítás

Az SRU réteg felelos a csomópontok tagságát nyilvántartani. A tagság vektor annyi bitet tartalmaz, ahány csomópont van a rendszerben. Minden idorésben ellenorzésre kerül az éppen küldo csomópont tagsága. Ha a redundáns fizikai csatornák közül legalább az egyik csatornán hibátlan keret vétele történik meg, akkor a küldo csomópont muködése hibátlannak tekintett az adott idopontban. A tagság érvényessége az adott csomópont következo idoréséig áll fenn. Az is következik ebbol, hogy amennyiben egy csomópont meghibásodott, akkor ez csak a következo küldési idorésében detektálható, azaz a tagság felbontása a TDMA ciklus ido. Ennek megfeleloen egy megjavított csomópont újraintegrálódása a rendszerbe legalább egy TDMA ciklus-t igényel. Ha egyetlen hibátlan CRC kódú üzenetet sem vesz a vevo csomópont, akkor törli a tagság vektorban a küldo csomópontnak megfelelo bitet.
Ha egy bizonyos csomópont nem vett egyetlen hibátlan üzenetet sem a küldo csomóponttól, akkor ezt úgy értékeli, hogy a küldo csomópont meghibásodott. Azonban ha az összes többi csomópont legalább egy hibátlan üzenetet vett a küldo csomóponttól, akkor különbözo eredményre jutnak a tagság vektorral kapcsolatban. Két klikk (clique) alakul ki a klaszteren belül, akik nem tudnak kommunikálni egymással, mivel különbözik a tagsági vektoruk és emiatt a C-állapotuk is. A TTP/C protokoll része egy mechanizmus, ami ilyen esetekben biztosítja, hogy a többség által látott állapot kerekedik felül, és a meghibásodott bemenetu csomópont tagsága törlodik.

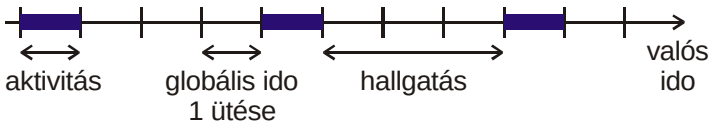
Óra szinkronizáció

Az idovezérelt protokoll része a lokális órák egy hibaturo belso szinkronizációja, melynek eredménye egy ismert pontosságú globális ido. Mivel minden vevo csomópont ismeri elore a várható érkezését minden üzenetnek, ezért az üzenet vételének valódi és a priori elvárt idejének különbsége jellemzi a küldo és a vevo csomópont órái közti különbséget. Ezt kihasználva nem szükséges külön sávszélességet fenntartani az órák szinkronizálására.

A globális ido és az óra szinkronizáció

Az ido fogalma rendkívül fontos a természettudományok területén. Sok természettudományos modell esetén az ido egy független bemeneti változó, ami meghatározza a rendszer állapotok sorozatát. Néhány alapveto fizikai konstans meghatározása is az ido egységén alapszik, a fizikai másodpercen.
Egy jellegzetes valós-ideju alkalmazásban az elosztott számítógépes rendszer nagy mennyiségu feladatot hajt végre párhuzamosan, például érzékelok mintavételezését, szabályozási algoritmusok végrehajtását, a felhasználó számára

információkat jelenít meg. Általában ezek a feladatok párhuzamosan, egyidoben, és különbözo csomópontokon hajtódnak végre. Ezen kívül a rendszerben még lehetnek replikált csomópontok is, vagy egyéb módon redundáns csomópontok a hibaturés növelése céljából. Azért, hogy garantálható legyen a teljes elosztott rendszer konzisztens muködése, biztosítani kell, hogy minden csomópont minden eseményt konzisztens módon dolgozzon fel, lehetoleg az események elofordulásának sorrendjében. A globális ido segít az idobeli konzisztencia fenntartásában, például idobélyegek használatával.
A valós ido egy suru idotengely mentén halad elore, végtelen számú idopillanatot tartalmazva. Az idotartam az idotengely egy intervalluma, melyet két idopillanat határol. Egy történést, ami egy idopillanatban végbemegy, eseménynek nevezünk. Ebbol következik, hogy a világ állapota szintén egy esemény. Az esemény idobélyege az esemény bekövetkezésekor fennálló globális ido állapotának azonnali rögzítése. A globális idot egy hibaturo belso óra szinkronizációs algoritmus biztosítja a TTA-n belül.
Az ido múlásának mérésére fizikai órákat használunk. A fizikai óra tartalmaz egy számlálót és egy fizikai oszcillációs mechanizmust, ami periodikusan egy eseményt generál, ami hatására a számláló értéke növelésre kerül. Ezt a periodikus eseményt mikroütés-nek (microtick) nevezzük. Két mikroütés között eltelt ido adja meg az óra szemcsézettségét, granularitását. Ezzel szemben az óra ütés, amit a TTP/C használ ido mérésre, a makroütés. A makroütés idotartama felhasználó által beállítható, és bizonyos számú mikroütésbol áll. Például minden 15-ik mikroütés felel meg egy makroütésnek. Ennek megfeleloen a rendszerben lévo csomópontokon akár eltéro mikroütést biztosító órák is lehetnek, de ettol még képesek azonos makroütések generálására.
Mivel lehetetlen az órákat tökéletesen szinkronizálni, ezért mindig fennáll a lehetosége annak, hogy a következo sorrendben következnek be az események: az egyik csomópont órája üt, egy esemény bekövetkezik, másik csomópont órája üt. Ebben az esetben az idobélyegek közötti különbség pontosan egy ütés. A tanulság az, hogy az órák véges felbontása miatt nem lehetséges az események konzisztens sorba rendezése az idobélyegek alapján. Ezt a problémát a TTA-ban a ritka idoalap bevezetésével oldják meg. A ritka-ido modellben az ido folytonosságát két részre osztjuk: egy aktivitás részre és egy hallgatás részre. Az aktivitás rész idotartamának nagyobbnak kell lennie az óra szinkronizációs algoritmus pontosságánál.

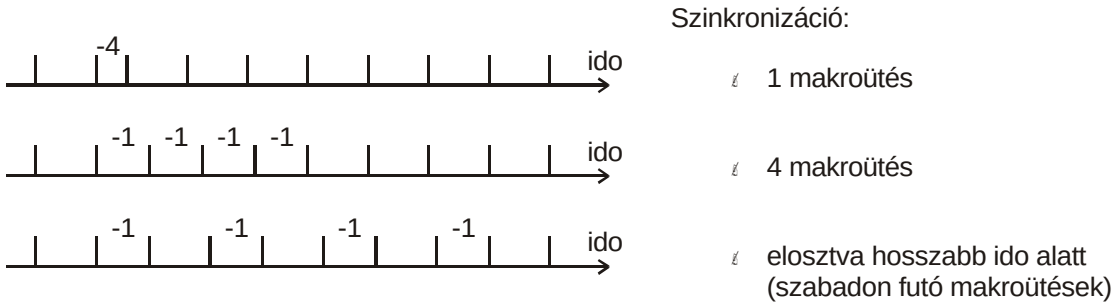


Ábra 7. Ritka idotengely

Az idobeli sorrendiség szempontjából minden esemény, ami az aktivitás idointervallum alatt történik meg, egyidoben történt eseménynek tekintett. Tehát azon események, amelyek az elosztott rendszer különbözo csomópontjain az azonos globális óraütés alatt következnek be, egyidejueknek feltételezettek. A különbözo aktivitás idointervallumokban bekövetkezo események (ha a szükséges hallgatási ido is elválasztja az aktivitásokat) már idoben sorba rendezhetok a globális idobélyegük alapján. Az architektura feladata azt biztosítani, hogy a fontos események, mint például az üzenetek küldése, az aktivitás ideje alatt történnek. Azon események idobélyegeit, amelyek nem állnak az elosztott rendszer irányítása alatt (és ezért a suru idotengelyen következnek be) valamelyik aktivitás idotartamhoz kell rendelni egy megállapodás szerint.

Óraszinkronizáció

A TTA rendszerben minden kommunikációs vezérlo mindkét csatornán a küldo idorése legelején kezdi el az adatátvitelt. Azonban a csomópontok órái nem tökéletesen szinkronizáltak, így minden vevo azt fogja tapasztalni, hogy a küldo nem a pontosan a meghatározott idoben kezdte el az adatátvitelt, hanem egy kicsit korábban vagy kicsit késobb. Ezért egy ún. fogadási idoablakot nyit a fogadó csomópont az elvárt küldési idopont környezetében. Mindegyik idorésben mindegyik vezérlo megméri a keret várt értkezési ideje és a valóságos érkezési ido közti különbséget. Egy globálisan kijelölt idotartamban, az újraszinkronizálási intervallumban, minden vezérlo kiszámítja az utolsó 4 mérés hibaturo átlagát, ami egyenlo lesz az óra állapot korrekcióval (clock state correction term). A hibaturo átlag képzés során eloször a legkisebb és a legnagyobb értéket, eldobjuk, majd az így kapott számok átlagát számítjuk ki. Az így kapott korrekciót arra használjuk, hogy a vezérlo óráját az átlaghoz igazítsuk, a makroütés generálása a korrekciós érték figyelembe vételével történik. Az igazítás során a makroütést késleltetjuk vagy gyorsítjuk, így a makroütések utolérik vagy bevárják az átlagot, majd a makroütések generálása a korábbi módon folytatódik. A makroütések közelítoleges folytonossága úgy biztosítható, hogy a korrekciós értéket szétosztjuk több makroütés között, és nem egy makroütés alatt próbáljuk az órákat szinkronizálni. Ez persze lassabb szinkronizációt eredményez.



Ábra 8. Makroütés szinkronizáció, ha az óra korrekció értéke -4

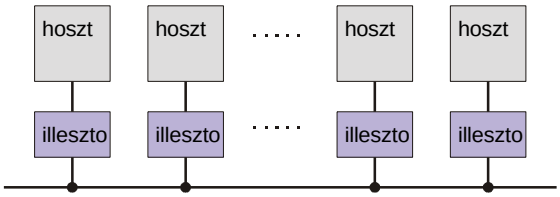
Folyamatosan végrehajtva a szinkronizációt a kommunikációs vezérlok a lokális óráikat az átlagos makroütéshez igazítják. Ezzel azonban még nem biztosított, hogy az órák az abszolút idovel is szinkronban legyenek. A rendszer óra csúszhat a külvilághoz képest. Az abszolút idohöz képesti csúszást a lokális oszcillátorok különbözo csúszása szabja meg. A külso világhoz történo szinkronizációt az alkalmazói szinten számított külso óra szinkronizációval biztosítható, ha erre szükség van.
További problémát okozhat a véges jelterjedési sebesség miatt a vezetéken létrejövo jelkésleltetés. Ha az órák közti pontos különbséget akarjuk meghatározni, akkor a jelkésleltetéseket is figyelembe kell venni. Nem csak a vezetékek, hanem a vonalmeghajtó áramkörök is okoznak késleltetést, így több korrekciós faktort kell felhasználni. Ezen felül még maga rendszer is használhat lassú egységeket, mint például csillagcsatolókat, ami további jelkésleltetést okoz. Az üzenetleíró táblázat tartalmaz egy mátrixot, ahol a különbözo csomópontok közötti jelkésleltetés megadható. Minden csomópontnak tudnia kell, hogy mennyit vonjon le a számított korrekciós értékbol a jelkésleltetések miatt.

Implementációk összehasonlítása

A következokben négy busz szerkezetet vázolunk, amelyek megfelelnek az elozo részekben kiemelt követelményeknek, valamint megoldást kínálnak bizonyos tervezési kihívásokra is. A négy architektúra közül kettot repülogép-ipari alkalmazásokra fejlesztették ki, a másik kettot meg elsosorban autóipari alkalmazásokra. Gazdasági okokból az autóipari alkalmazások esetében fontos szempont az alacsony ár (több milliós példányszám). Mind a négy busz a rendszerek idovezérelt megközelítését használja, hibaturo elosztott óra szinkronizációt alkalmaz és busz orzoket, vagy más, egyenértéku mechanizmust használ a fecsegés (babbling idiot) hibával szembeni védelemre. A négy szerkezet különbözik hiba hipotézisében, alkalmazott mechanizmusában, szolgáltatásaiban, valamint garancia, teljesítmény és költsége szerint.

SAFEbus

A Honeywell által kifejlesztett SAFEbus TM (fo tervezoi: Kevin Driscoll és Ken Hoyme) a Boeing 777 Airplane Information Management System (AIMS) magjaként szolgál, mely számos kritikus funkciót, példának okáért a pilótafülke muszerfal kijelzését támogatja. A busz ARINC 659-ként lett szabványosítva és a Honeywell-i kivitelezés variációit használják, ill. mérlegelik használhatóságát más légi- és urtechnikai berendezéseknél. A szerkezet a ábrán bemutatotthoz hasonló busz topológiát használ; busz illesztoi (melyek neve: Bus Interface Unit, vagy BIU) duplikáltak és a kommunikációs busz négyszeresen redundáns.



Ábra 9. SAFEbus topológia (redundancia jelzése nélkül)

A SAFEbus funkcionalitása leginkább a busz illeszto egységekben valósul meg, amik óra szinkronizálási, üzenet-rendezo és továbbító funkcióval rendelkeznek. Egy pár minden egyes busz illeszto egysége egy különálló potenciálisan meghibásodható egység, és párjának busz felügyelojeként is muködik, annak az összeköttetési médiához való hozzáférését ellenorizendo.
Ugyan egy pár minden egyes busz illeszto egysége más fizikai buszt használ, de mégis mind a négy olvasására képes; maguk az összeköttetést megvalósító buszok egyenként két adat- és egy óra vezetéket tartalmaznak és 30 MHz-n muködnek. A busz vezetékek és meghajtóik a ’vagy’ kapuk elektromos tulajdonságaival rendelkeznek (azaz amennyiben több, különbözo busz illeszto egység ugyanabban az idopontban ugyanazt a vezetéket hajtja meg, a keletkezo jel a különbözo bemenetek ’vagy’ kapcsolata). Néhány része a protokollnak kihasználja ezt a tulajdonságot; különösen az óra szinkronizáció az, amely megvalósítható esemény alapú algoritmus használatával.

A páros busz illeszto egységek a küldonél és a fogadónál, és a négyszeresen redundáns buszok elég redundanciát biztosítanak a SAFEbus számára, hogy az kölcsönös és állandó üzenetközvetítést tegyen lehetové (a Honeywell féle kivitelezésben), a már Davies és Wakerly által felvázolt megközelítéshez hasonló módon (ez a figyelemreméltó tanulmány évekkel elore látta a Bizánci hibaturés számos problematikus pontját és azok megoldását). A szerkezet, mindemellett, hogy felhasználói szintu hiba toleranciával rendelkezik (önellenorzo párok) a meghibásodott egységet automatikusan és gyorsan helyettesíti a tartalék egységgel.
Hiba hipotézise véletlenszeru hibákat tartalmaz, különbözo csomópontokon felmerülo hibákat (csomópontonként egy hiba), valamint magas arányú hiba bekövetkezést. Sohasem adja fel és pontosan meghatározott újraindító és felépülo stratégiával rendelkezik a hiba hipotézist meghaladó hibák esetére. Az AIMS alkalmazásának térbeli közelségi hibáit az egész rendszer duplikálásával tolerálja. A SAFEbus-t utasszállító repülogépekben való használatra hitelesítették, a Boeing 777-es esetében rendelkezik szélesköru tapasztalattal. A Honeywell kivitelezést saját fejlesztorendszer teszi lehetové.
A SAFEbus az itt tárgyalt buszok közül a legfejlettebb, az, amelyik a legkevesebb kompromisszumot is köti egyben. Mivel minden egyes összetevoje párosított (és buszához külön óra- és adatvezeték szükséges), a kereskedelmi használatra rendelkezésre állók közül a legköltségesebb is (általában néhány száz dollár csomópontonként).

TTA

Az idovezérelt architektúrát (Time Triggered Architecture, TTA) a Bécsi Muszaki Egyetem professzora, Hermann Kopetz, és kollégái fejlesztették ki. A szerkezet kereskedelmi fejlesztését a TTTech látja el és jelenleg Audi és Volkswagen típusú személygépkocsikba telepítik leginkább biztonság-kritikus részeken, valamint a Honeywell által gyártott repülogép motorok repülés-kritikus funkciói részeként.
A TTA jelenlegi megvalósítása busz topológiát használ. Illesztoi a TTP/C protokollt implementálják, ami a TTA szíve és óra szinkronizációt, valamint üzenet átvitelt tesz lehetové. Az összeköttetési busz kettozött és mindegyik vezérloje mindkettot meghajtja a részlegesen független busz orzokön keresztül. A TTA egy átlagolásos óra szinkronizációs algoritmust használ, Lundellius és Lynch alapján. Ezen algoritmus a vezérlokben található, ám ahhoz túlzottan forrásigényes, hogy a busz felügyelokben is replikálva legyenek. Az orzok független órákkal rendelkeznek, ezért szükségük van egy keret kezdet jelzésre a vezérloiktol. Ez bizonyos értelemben szukíti függetlenségüket (áramellátásuk, valamint más forrásaik is közösek a vezérloikkel), így a TTA jövobeni megvalósulásai a csillag topológiához hasonlóval lesznek felszerelve. Ebben az esetben a busz orzo funkció a csillag csatolón található, így teljesen független a vezérloktol: a csillag csatolók és a vezérlok különálló egységeket alkotnak. A csatolók a hibaturés végett kettozöttek és különbözo helyeken vannak elhelyezve, hogy ellenálljanak a térbeli közelségi hibáknak. Jel átalakítást is alkalmaznak a gyengén specifikáción kívüli hibák számának csökkentésére.
A TTA a csoportos tagságra és a klikkesedés elkerülésére algoritmust használ; ez teszi lehetové óraigazító algoritmusa számára a többszörös hibák tolerálását (újrakonfigurálás útján, ami kizárja a meghibásodott tagokat) és összekapcsolva az ellenorzo összegek használatával (amelyek digitális aláírásként foghatók fel) mindez kölcsönösen folyamatos üzenetközvetítési formát eredményez. Az ajánlott kiterjesztések olyan formában nyújtanak állapot gép replikációt, hogy az transzparens az alkalmazás számára.
Hiba hipotézise tetszoleges hibát tartalmaz, különbözo csomópontokban felmerülo hibákat (csomópontonként egy hiba), abban az esetben, ha ezek legalább ketto TDMA kör távolságra jelentkeznek egymástól (ez teszi lehetové a tagsági algoritmusnak a hibás csomópontok kizárását). Sohasem adja fel és pontosan meghatározott újraindító és felépülo stratégiával rendelkezik a hiba hipotézist meghaladó hibabeérkezésekbol.
Az elkészült TTA prototípusokat szélesköru tesztelésnek és hiba injektálásnak vetették alá, kísérleti jármuvekben is. Algoritmusai közül több formálisan is beigazolódott és a még fejlesztés alatt álló repülogépi alkalmazások a tervek szerint az FAA hitelesítést is maguk után vonják a jövoben. Kiterjedt eszköztára megfelel a standard CAD környezethez (azaz: Matlab/Simulink és Beacon). Jelenlegi formájában 25 Mbit/s átviteli sebességgel felszerelt, de már kutatási projektek foglalkoznak a gigabites kivitelezéssel. A TTA vezérlok és a csillag csatolók (amelyek lényegében módosított vezérlok) viszonylag egyszeruek és olcsón eloállíthatóak nagy mennyiségben.
Az itt tárgyalt szerkezetek között egyedülálló a TTA, mivel mind a személygépkocsi gyártásban – ahol a nagy mennyiségben való eloállítás alacsony árakat eredményez -, mind pedig a légi jármuvek esetében szükséges mélyreható biztonsági vizsgálatot lehetové teszi.

SPIDER

A „Scalable Processor-Independent Design for Electromagnetic Resilience (SPIDER) rendszer fejlesztoi, Paul Miner és munkatársai, a NASA Langley Research Centerben dolgoznak egy olyan kutatási program keretében, amely sugárzás által gerjesztett hibák (HIRF/EMI) újraélesztési stratégiáit keresi, és ami felhasználható esettanulmányként újabb kivitelezésu légi elektronikus hardware-ek biztonsági irányelveinek tesztelésekor (DO 254).
A SPIDER csillag konfigurációt használ, amelyben az illesztok elhelyezhetok hosztjaikkal együtt a központi csatoló egységben, ami szintén tartalmaz aktív részeket, ún. redundancia intézo egységeket (Redundancy Management Unit, RMU).
Az óra szinkronizáció, valamint a SPIDER más szolgáltatásai új elosztott algoritmusok útján érhetok el, melyek a busz illeszto egységek és a redundancia intézo egységek között valósulnak meg. A szolgáltatások közé tartozik az interaktív, folytonos üzenetközvetítés, a hibás csomópontok azonosítása (amelybol a tagság szolgáltatás már könnyen szintetizálható). A SPIDER hiba hipotézise hibrid hiba modellt használ, ami korlátlan számú hibát tartalmaz, valamint

bizonyos számú többszörös hibakombinációt is engedélyez. Algoritmusai újak, rendkívül hatékonyak és formális igazolásuk is folyamatban van.
A SPIDER egy valóban érdekes tervezés, mely a többi, itt tárgyalt busztól eltéro topológiát és algoritmust használ. Mégis, hiszen még mind a kivitelezésében, mind a bevezetésében folyamatban lévo kutatási projektrol lévén szó, a rendszer a kereskedelmi termékekkel direkt módon nem összehasonlítható.

FlexRay

A BMW, DaimlerChrysler, Motorola és Philips cégeket magába foglaló konzorcium autóipari vezérlések céljából fejleszti a FlexRay-t. Az itt tárgyalt többi busztól abban különbözik, hogy muködése az idovezérelt és az esemény-vezérelt tevékenységi körök között oszlik meg. A FlexRay protokoll eddig megjelent leírásai még vázlatosak.
A FlexRay vagy az aktív csillag összeköttetési elrendezést használja, vagy a passzív busz topológiát. A fizikai média kettozése mindkét esetben opcionális. A FlexRay csillag konfigurációja (csakúgy, mint a TTA-é) olyan elosztott konfigurációkban is alkalmazható, ahol az alrendszerek linkekkel kapcsolódnak össze. Mindegyik FlexRay illeszto (nevük kommunikációs vezérlo), az illesztovel együtt megtalálható, külön busz orzokön keresztül hajtja meg a fizikai médiát jelento vezetékeket. (Ez lényegében azt jelenti, hogy a két busz használatával mindegyik csomópontnak három órája van: egy a kommunikációs vezérlonek, egy-egy pedig a busz orzoknek; ez más, mint a TTA konfigurációja, ahol a vezérlo egy órája mellett a két busz orzo is egy órán osztozik.) A TTA busz konfigurációjához hasonlóan a FlexRay busz orzoi sem teljesen függetlenek a kommunikációs vezérloktol.
A FlexRay nagyobb rugalmasságra törekszik, mint a többi itt tárgyalt busz; ez a cél magában az elnevezésben is megmutatkozik. Mint azt már említettük, ennek a rugalmasságnak egyik megnyilatkozása az idovezérelt és az esemény-vezérelt muveletek kombinációja. A FlexRay minden idociklust részekre bont, egy statikus idovezérelt, és egy dinamikus esemény-vezérelt részbe fordulnak. A két rész közötti elválasztás beállítása a tervezési idoben történik, és az eredmény a vezérlokön, valamint a busz orzokön kerül tárolásra. A ciklus esemény vezérelt része a Byteflight protokollt használja. A SAFEbus-tól és a TTA-tól eltéroen a FlexRay nem tárolja az idovezérelt rész teljes ütemezését minden vezérlon. Helyette, a ciklus idovezérelt része számos fix méretu résre osztódik, és mindegyik vezérlo és busz orzoje csak a saját adatátviteléhez rendelt résrol rendelkezik információval (a nagyobb sávszélességet igénylo csomópontok több réshez vannak hozzárendelve, mint a kevesebbet igénylok). A vezérlok csak a rendszer indulásakor tudják meg a teljes ütemezést. Mindegyik csomópont csatolja azonosítóját az elküldött üzeneteihez; induláskor a csomópontok ezeket az azonosítókat használják bemeneti tárolóik megjelölésére, ahogy az ütemezés körvonalazódik (például: amennyiben az 1-es és 7-es résben érkezo üzenetek azonosítója 3-as, attól kezdve minden csomópont elszállítja az 1-es és 7-es buffer tartalmát ahhoz a taszkhoz, amelyik a 3-as csomópont által küldött üzenetekkel foglalkozik. Ez a pont sebezhetonek tunik, hiszen egy hibás csomópont feltunhet úgy, mintha egy másik volna (vagyis rossz azonosítóval küldi el az üzenetet) induláskor, ilyen módon elrontva a adatátvitel fennmaradó részét. Még nem tisztázott, hogy ennek a hibának az elhárítása milyen módon történik.
A TTA-hoz hasonló módon a FlexRay is a Lundelius-Welch óraszinkronizációs algoritmust használja, mégis, a TTA-tól eltéroen nem használ tagság ellenorzo algoritmust a hibás csomópontok kiküszöbölésére. A FlexRay nem rendelkezik best-effort (megteszi, ami tole telik) jellegu üzenetszállítást biztosító szolgáltatásokkal; egészen pontosan nem rendelkezik interaktív, konzisztens üzenet átvitellel. Ez azt jelenti, hogy minden hibaturési mechanizmussal el kell látni az alkalmazást. A FlexRay-rol eddig megjelent írások nem határozzák meg pontosan a hiba hipotézisét, és egyelore úgy tunik, bizonyos hibák elhárítására nem rendelkezik megfelelo mechanizmussal (pl.: SOS hibák, vagy az aszimmetrikus átvitel más hibái). A „sose add fel”-stratégia ismertetése még nem készült el, sem pedig a biztonságosság szisztematikus vagy formális megközelítései.
A FlexRay érdekessége az idovezérelt és eseményvezérelt rendszerek muködéseinek vegyítésében rejlik, potenciális jelentosége pedig kifejlesztoik ipari befolyásában rejlik. Jelenleg ez a kereskedelmi buszok közül a leglassabb, melynek adat átviteli sebessége nem több, mint 10 Mbit/s.

Összefoglalás és értékelés

A biztonság-kritikus busz szerkezetek olyan tulajdonságokkal és szolgáltatásokkal bírnak, amelyek hozzásegítenek biztonság-kritikus rendszerek létrehozásához. Mint bármely más rendszerváz, ezek a buszok is választást kínálnak a rendszer fejlesztoinek: koherens szolgáltatások halmazával rendelkeznek, eros tulajdonságokkal és magas szintu kivitelezéssel, mégis a fejlesztoknek fel kell áldozniuk a tervezési szabadság egy részét a szolgáltatások teljes kihasználhatósága érdekében. Például, a buszok mindegyike idovezérelt megközelítést használ, és a rendszer fejlesztoi a megadott kereteken belül kell kiépíteniük saját alkalmazásaikat. Cserébe, a buszok eros particionálás biztosítására képesek: a különálló alkatelemek vagy alkalmazások (repülésügyi terminológiában: függvények) hibái nem tudnak átterjedni, és az egész busz muködését sem tudják elrontani (hiba hipotézisük határain belül).
A particionálhatóság valójában alapveto követelmény. Biztosítja, hogy egy hibás függvény ne veszélyeztessen másikakat, mégis sok biztonság-kritikus rendszer estében akár egyetlen függvény hibája is végzetessé válhat, így a különálló függvényeknek önmagukban hibaturoeknek kell lenniük. Következésképpen a buszok többsége rendelkezik olyan mechanizmussal, ami segíti a hibaturo alkalmazások fejlesztését. Kulcsfontosságú követelmény ebben az esetben a következetes üzenet-átvitel: ez által biztosított, hogy mindegyik egység és tartalékja, vagy a szavazás minden tagja konzisztens állapotban maradjon. Az itt tárgyaltak közül három busz rendelkezik ezzel az alapveto szolgáltatással; némelyikük ezt más szolgáltatásokkal együttesen teszi, például a normál/tartalék váltással vagy a tagsági szolgáltatással, ami alacsony szinten megvalósítva kivitelezheto nagymértékben növelt hatásfokkal, és nagymértékben csökkentett látenciával. A FlexRay az egyetlen, amely nem rendelkezik ezekkel a szolgáltatásokkal, melyek hiányában minden

egyes hibaturési mechanizmusa csak az alkalmazói programban valósítható meg. Ily módon, olyan alkalmazói programozók, akik kevés tapasztalattal rendelkeznek a hibaturo rendszerek sajátosságai terén, vállalnak felelosséget a hibaturo rendszer tervezése, kivitelezése és a biztonság területein. Ez nem csupán a költségeket és a pontos munka ellenorzésének nehézségeit növeli, hanem egyben a számítási költségeket és a látencia mértékét is. Például az interaktív, állandó üzenet-közvetítés hiányában az alkalmazói programoknak explicit módon kell továbbítani a kereszt-összehasonlítások többszörös köreit, amelyek a szolgáltatás magasabb szintu megvalósulásához kellenek, és az üzenetmennyiség jelentos növelését okozzák.
Nem valószínu, hogy bármelyik egyedülálló busz szerkezet ki tudna elégíteni minden igényt és piacot, így lehetséges, hogy a FlexRay-re jellemzo felhasználói szintu hibaturo szolgáltatások hiánya bizonyos területeken tetszést kelt majd. Az is könnyen elképzelheto, hogy az új, ill. módosított szerkezetek új piacok és igények kielégítésére lesznek alkalmasak. (Például ajánlott a TTA-nak szintén támogatnia a FlexRay-hez hasonlóan az esemény-vezérelt és idovezérelt kommunikációs képességet, meghatározott idorések elhelyezése révén a CAN-t szimulálni. A szimulálás tulajdonképpen gyorsabb, mint az igazi CAN busz, és közben a TTA összes biztonsági jellemzoje megmarad).

Referenciák

H. Kopetz, G. Bauer , “The Time-Triggered Architecture”, IEEE Special Issue on Modeling and Design of Embedded Systems, October 2002.
R. Maier, G. Bauer, G. Stöger, S. Poledna, ”Time-Triggered Architecture: A Consistent Computing Platform”, IEEE Micro, July 2002.
H. Kopetz , ”Real-Time Systems: Design Principles for Distributed Embedded Applications”, Kluwer Academic Publishers, ISBN 0-7923-9894-7, 1997.
J. Rushby, ”A Comparison of Bus Architectures for Safety-Critical Embedded Systems”, Final report for SRI project 6464, SRI International, 2001.