A konformancia- és együttműködés-tesztelés bemutatása

KRÉMER PÉTER

Ericsson Magyarország Kft., Test Competence Center
Peter.Kremer@ericsson.com

Kulcsszavak: együttműködés-tesztelés, konformancia-tesztelés, ROHC

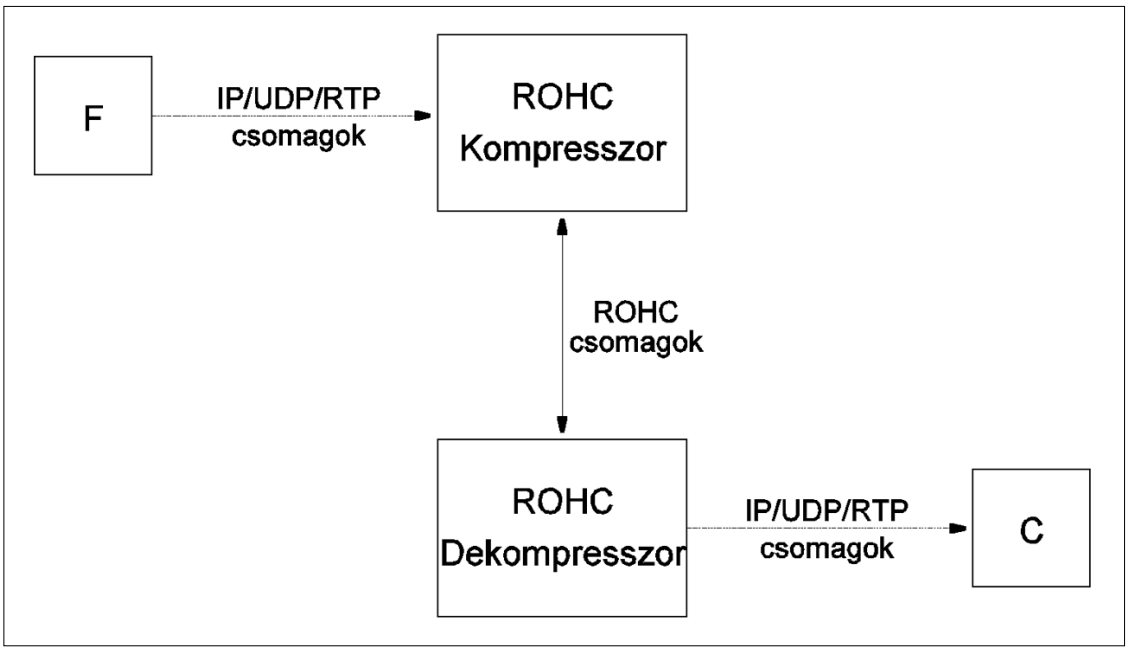
A cikk elsődleges célja bemutatni az együttműködés-tesztelés folyamatát és fajtáit. Emellett azonban kitér a konformancia-tesztelésre is. Egyrészt azért, mert azon keresztül mutatja be az együttműködés-tesztelést. Másrészt pedig azért, mert manapság mindkét tesztelési mód fontossá vált és nem lehet csak az egyik vagy csak a másik módszer alapján megalapozott véleményt formálni egy-egy berendezésről, protokoll megvalósításról. A könnyebb érthetőség érdekében néhány konkrét példát is bemutatok a ROHC protokoll tesztelésére.

1. Bevezetés

A két tesztelési módszer közül a konformancia teszt a régebbi, ezt a fajta tesztelést távközlési berendezések gyártói kezdték el használni annak bizonyítására, hogy a termékük megfelelően működik. Ez a módszer a mai napig nemcsak használatban van, de igen népszerű is. A bonyolult és drága berendezések miatt a távközlés világában nagyon fontos, hogy minden pontosan meghatározott módon történjen. Ez a megközelítés nem csak a távközlési szabványokban tükröződik (alapos, precíz leírás; jól definiált interfészek), hanem a berendezések teszteléséhez használt módszerekben is. Ezért használják az aprólékos, precíz konformancia-tesztelést.

Az együttműködés-tesztelés az IP protokollok fejlődésével kapott és kap jelenleg is egyre nagyobb szerepet. Ez a tesztelési módszer ugyanis az IP protokollok esetén a leggyakrabban használt eszköz a különböző implementációk ellenőrzésére. Az IP alapú protokollok egyszerű, olcsóbb eszközökön működnek, az ilyen berendezéseknek sokkal több gyártója és vevője van. Ebből következően a berendezések tesztelésénél is teljesen más a cél, így nyilvánvaló, hogy teljesen másfajta módszert kell használni a tesztelésnél is. Az IP-s világban elterjedt szabványokban általában nincsenek olyan jól definiált interfészek, mint a távközlési szabványokban. Ezen kívül gyakran előfordulnak olyan esetek, ahol az implementációra bíznak egyes döntéseket vagy egyszerűen csak nem definiálják az elvárt működést. Ehhez a megközelítéshez pedig az együttműködés-tesztelés áll közelebb.

A következő fejezetben bemutatjuk a ROHC protokollt, amelyet a továbbiakban példaként fogok használni. Ezután röviden bemutatom a konformancia-tesztelést, illetve a ROHC protokoll teszteléséhez használt teszt-elrendezéseket. A cikk további részeiben pedig az együttműködés-tesztelés részletes ismertetése következik.



1. ábra
Tipikus ROHC elrendezés

2. A ROHC protokoll rövid ismertetése

A ROHC (RObust Header Compression) protokoll IP csomagok fejlécének tömörítésére szolgál, pont-pont jellegű összeköttetések esetén. Amikor IP protokollt használunk vezeték nélküli hálózatokban, akkor a legnagyobb probléma általában az IP fejléc nagy mérete a hasznos adathoz képest. Beszédátvitel esetén a hasznos rész 15-20 bájt körül van, míg a fejléc (IPv4, UDP és RTP protokollok használata esetében) 40 bájt hosszú. A szűkös sávszélesség minél hatékonyabb kihasználásához tehát valamilyen tömörítésre van szükség.

A ROHC tehát egy olyan IP fejléctömörítő megoldás, amit kifejezetten vezetéknélküli hálózatokhoz fejlesztettek ki. Más tömörítő eljárásoktól eltérően nem az egy IP csomagon belüli redundanciát használja ki, hanem az egymást követő – de ugyanahhoz az adatfolyamhoz tartozó – csomagok közötti hasonlóságot. Egy adatfolyam esetében az IP cím például nem változik, így ezt az információt elegendő egyszer átküldeni a csatornán. Más esetekben egy mező értéke – a többi mező értékének ismeretében – kiszámolható (például a csomag hossza), így ezeket sem kell minden esetben továbbítani.

Látható tehát, hogy az IP csomagok tömörítéséhez és visszafejtéséhez nemcsak az adott IP csomag tartalmának ismerete szükséges, hanem az azt megelőző csomagoké is. Ez különféle adatbázisok és táblázatok létrehozását és folyamatos frissítését igényli mindkét oldalon. A ROHC hatékonyságára jellemző, hogy a 40 bájtos fejléc helyett mindössze 1 bájtban képes az eredeti csomag visszaállításához szükséges minden információt eltárolni – és ebben már az átviteli hibák elleni CRC védelem, valamint a ROHC csomag típusának megjelölése is benne van!

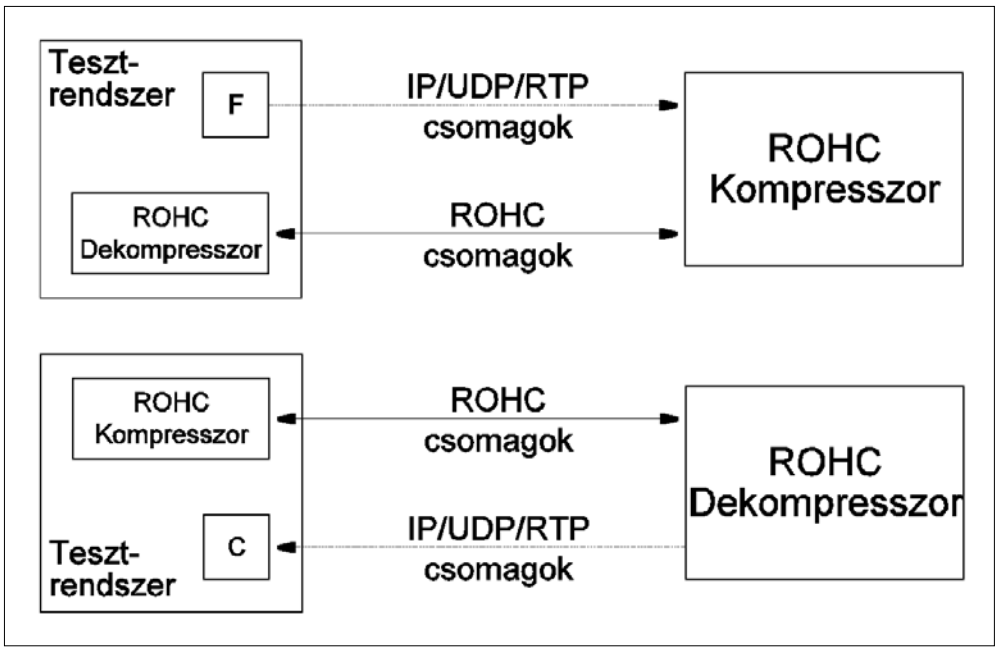
Az *1. ábrán* egy tipikus ROHC elrendezésre láthatunk példát. Az ábrán **F**-el jelöljük a forrás, és **C**-vel a célállomást. A forrásból jövő csomagokat a ROHC Kompresszor az aktuális állapotának megfelelően tömöríti. Az eredeti csomagok a ROHC Dekompresszor visszaállítja és elküldi a célállomásnak. A forrás és célállomás számára az eljárás teljesen átlátszó és semmilyen megkötést nem tartalmaz.

3. Konformancia-tesztelés

Konformancia-tesztelés során azt ellenőrizzük, hogy egy protokoll implementáció megfelel-e a szabványnak. Távközlési berendezések és protokollok esetében ez a tesztelési mód az elfogadott. Vannak olyan szabványosítási testületek (pl. ETSI, ITU, 3GPP), amelyek konformancia-tesztsorozatokat is készítenek, illetve szabványosítanak – ezeket egy szintén szabványos tesztleíró nyelven adják ki. A berendezések gyártói pedig ezen szabványos tesztsorozatok segítségével tesztelik termékeiket és ellenőrzik, hogy azok megfelelnek-e a vonatkozó szabványoknak.

Konformancia-tesztek során az implementáció belső működését nem ismerjük, ahhoz csak a specifikációban meghatározott interfészeken keresztül, protokollüzenetek felhasználásával férünk hozzá. Ahhoz, hogy egy protokollhoz könnyű legyen eszköz-független konformancia-teszteket írni, a külső interfészek pontos definiálása, valamint az interfészeken keresztül elérhető szolgáltatások minél szélesebb köre szükséges (például a protokoll különböző paramétereinek beállítása). Mivel a konformancia-teszteket szabványos (vagyis minden implementáción egyforma) interfészeken keresztül végezzük, ezért lehetnek a konformancia-tesztsorozatok implementáció-függetlenek. Ez azt is jelenti, hogy ugyanazt a tesztsorozatot változtatás nélkül futtathatjuk le különböző protokoll megvalósításokon.

Nagybonyolultságú berendezések esetében előfordul, hogy együttműködési teszteket is elvégeznek, de csak a konformancia-tesztek után. Ezeket az együtt-



2. ábra
Konformancia-tesztelés elrendezése kompresszor és dekompresszor tesztelése esetén

működési teszteket általában már a majdani vevő kérésére és annak hálózatában végzik. A cél ekkor annak megállapítása, hogy az adott eszköz képes-e együttműködni a meglévő berendezésekkel, amelyek általában több különböző gyártó termékei.

Konformancia-tesztelésben alapvetően háromféle teszttípust különböztetünk meg:

– normál viselkedés:
  a normál működés során fellépő eseteket vizsgáljuk, hibás üzeneteket nem küldünk;
– negatív viselkedés:
  az implementáció viselkedését vizsgáljuk olyan esetekben, amikor az szintaktikailag hibás üzeneteket kap;
– helytelen viselkedés:
  szintaktikailag helyes, de szemantikailag helytelen üzenetekre ellenőrizzük az implementáció viselkedését.

Természetesen nem tudunk minden lehetséges esetet megvizsgálni konformancia-tesztelés során, de a fenti három teszttípus mindegyike előfordul egy körültekintően megírt tesztsorozatban.

3.1. A ROHC protokoll konformancia-tesztelése

Konformancia-teszt esetében a kompresszort és a dekompresszort is külön-külön kell vizsgálnunk. Mindkét teszt-elrendezésre a *2. ábrán* láthatunk példát.

Kompresszor tesztelése esetén a tesztrendszerünk egyszerre viselkedik forrásként és dekompresszorként. A forrás előállítja az IP csomagokat, a dekompresszor segítségével pedig ellenőrizzük, hogy a vizsgált kompreszszor megfelelő ROHC csomagok állít-e elő. Az a ROHC csomag számít megfelelőnek, amely mind szintaktikailag, mind szemantikailag helyes és nem utolsó sorban az, amelyikből az eredeti IP csomag visszaállítható.

Dekompresszor-teszt esetében a tesztelő rendszerünk egy kompresszort és egy célállomást helyettesít. A dekompresszornak olyan ROHC csomagokat küldünk, amit egy jól működő dekompresszor képes feldolgozni és abból az eredeti IP csomagokat visszaállítani. A dekompresszor által rekonstruált IP csomagokat a tesztrendszer összehasonlítja az eredetivel (vagyis azzal, amiből a kompresszorunk a ROHC csomagokat előállította) és ebből megállapítja, hogy a tesztelt dekompresszor helyesen működik-e.

4. Együttműködés-tesztelés

Az együttműködés-tesztelés során azt ellenőrizzük, hogy ugyanazon protokoll két különböző megvalósítása képes-e egymással – a specifikációnak megfelelően – kommunikálni.

Fontos tulajdonsága az együttműködés-tesztelésnek, hogy nem egy, hanem egyszerre két implementációt vizsgál. Ha egy teszt során hibát észlelünk, akkor nem csak azt kell megállapítani, hogy mi okozta a hibát, hanem azt is, hogy melyik implementáció miatt történt a hiba. Mivel ezen implementációkat különböző cégek készítik (egy cégen belül ritkán készítenek több független implementációt), ezért érdekes módon a gyakorlatban nem az a legfontosabb kérdés, hogy mi volt a hiba, hanem az, hogy melyik cég terméke okozta azt.

Természetesen együttműködés-tesztelés után is maradhatnak még hibák az implementációkban. Előfordulhat például, hogy a specifikáció félreérthető vagy nem egyértelmű és mindkét implementációban ugyanazon a (hibás) módon valósítanak meg egy funkciót. Az ilyen fajta hibákat az együttműködés-tesztelés során nem mindig lehet felderíteni.

Az együttműködés-tesztelést azonban nem kizárólag csak protokoll-megvalósítások ellenőrzésére használják. Az IETF-ben (Internet Engineering Task Force – nemzetközi szabványosítási szervezet, amely internetes protokollokkal foglalkozik) magát a szabványosítás alatt álló protokollt is együttműködés-tesztelés során validálják. Egy protokoll specifikációt csak akkor fogadnak el szabványként, ha a protokoll két, egymástól független megvalósítása képes együttműködni. Vagyis a két implementáció együttműködés-tesztje tulajdonképpen magának a protokoll-specifikációnak egyfajta ellenőrzése is.

A bevezetőben már említettük, hogy az IP alapú protokollok esetében a specifikációk nem olyan részletesek, mint a távközlési protokollok esetében. Ennek következménye, hogy az egymásra épülő protokollok között nincsen olyan jól definiált, éles határ. Együttműködési tesztnél azonban nincs is erre szükség. Egyszerűen csak összekötjük a két berendezést és megfigyeljük, hogyan működnek. Az egyes üzenetek pontos megadása helyett a felhasználói interfészen keresztül utasítjuk a protokoll-implementációt a kívánt teszt (például kapcsolatfelvétel) lefuttatására. Ez az interfész általában implementáció-függő, tehát az adott rendszer alapos ismerete szükséges a tesztek végrehajtásához. Az együttműködés-tesztelés így egyrészt egyszerűbb – mert nem kell a protokollüzeneteket egyesével előállítani és elemezni, másrészt viszont bonyolultabb – mert csak az adott implementáció ismeretében lehet a teszteket végrehajtani.

A konformancia-tesztelésnél ismertetett háromféle teszttípus közül (normál, negatív, helytelen) az együttműködés-tesztelésnél általában csak egyetlenegy fajtát tudunk vizsgálni, ez pedig a normál működés tesztelése. Mivel a protokollüzeneteket egy – a szabványnak elvileg megfelelő – implementáció állítja elő, így a legtöbbször nincs lehetőségünk sem a negatív, sem pedig a helytelen viselkedés ellenőrzésére. A következő szakaszban láthatjuk majd, hogy hogyan lehet mégis ilyen eseteket is vizsgálni.

Az utóbbi időben érdekes tendencia figyelhető meg az együttműködés-tesztelés területén. Ahogyan egyre fontosabbá válnak az IP alapú protokollok, úgy az azok ellenőrzésére leginkább használt együttműködés-tesztelés is kezd egyre szervezettebbé válni. A szervezettség fokát tekintve az alábbi öt szintet különböztethetjük meg:

1. *Ad-hoc módon*

Ez a teljesen szervezetlen esetet jelenti, amikor mindenféle előzetes elképzelés és tervezés nélkül zajlik a tesztelés. A fejlesztők vagy megbeszélik előre, hogy tesztelni fognak vagy csak véletlenül találkoznak, például egy IETF megbeszélés után.

2. *Szervezett formában, segédeszközök nélkül*

Ebben az esetben már előre megbeszélik, hogy mikor fognak találkozni, és esetleg még tervet is készítenek arra, hogy a szabvány mely részeit fogják alaposabban leellenőrizni. A teszteléshez a szükséges alapvető környezetet (asztalok, székek, konnektorok, hálózati kapcsolat) a vendéglátó biztosítja.

3. *Szervezett formában segédeszközökkel*

Az előző szinttől abban különbözik, hogy itt az együttműködés tesztet szervezők az alapvető környezeten kívül olyan segédeszközöket is biztosítanak, amivel a tesztelés nemcsak könnyebb, de esetleg megismételhető is lesz (például ROHC esetében egy megfelelő forrás, amely képes ugyanazokat a csomagokat kiküldeni vagy megadott minta szerint változtatni).

4. *Informális leírás alapján*

A fentieken túl, ebben az esetben előre készítenek egy dokumentumot a végrehajtandó tesztekről. Ez tartalmazza az egyes tesztesetek részletes – szöveges – leírását a konfigurálástól a teszt közben végrehajtandó lépéseken keresztül egészen a teszt kiértékeléséig.

5. *Formális leírás alapján (automatikusan)*

Ez tulajdonképpen egy szabványos tesztleíró nyelven megírt tesztsorozat, amely az implementáció-függő paraméterek specifikálása után végrehajtható. A tesztsorozat összes tesztesete ilyenkor automatikusan – külső beavatkozás nélkül futtatható.

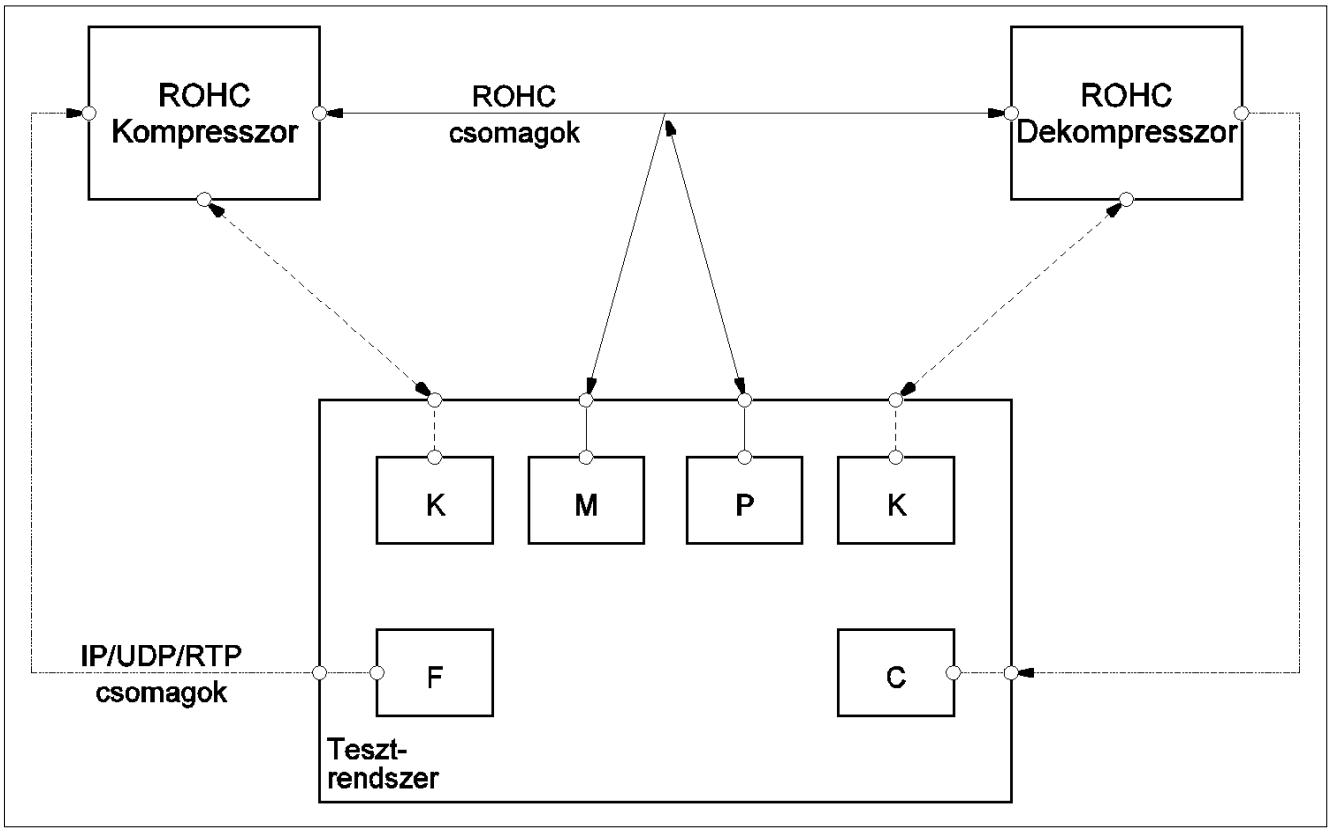
Ahogyan az egyes szinteken haladunk egyre feljebb, úgy válik a tesztelés egyre hatékonyabbá, vagyis a magasabb szinteken egyre kevesebb időre van szükség ugyanazon teszteset lefuttatására. A ROHC protokoll példájánál maradva, az első két szinten még az sem biztosított, hogy a forrás által kiküldött, illetve a célállomáshoz érkező csomagokat valamilyen módon rögzítsük. Ha pedig valahogy mégis sikerül rávenni az implementációnkat, hogy ezeket az adatokat elmentse, akkor ütközünk a következő problémába a különböző formátumú adatok összehasonlításával.

A konformancia-tesztelés már az 5. szinten, az együttműködés-tesztelés egyelőre a 2-3., nagyon ritkán pedig a 4. szinten helyezkedik el. Ez persze nem azt jelenti, hogy az egyik módszer jobb, a másik pedig rosszabb. Egyszerűen csak arról van szó, hogy az együttműködés-tesztelés még nem ért el arra a szintre, ahová a konformancia teszt. A fejlődés azonban egyértelműen látható, tehát már csak idő kérdése és az együttműködés-teszteket is automatikusan lehet majd végrehajtani.

4.1. A ROHC protokoll együttműködés-tesztelése

A *3. ábrán* az automatizált együttműködés-teszteléshez használt teszt-elrendezés látható. Az automatizált tesztek futtatásához a tesztrendszernek egy tesztleíró nyelven előre rögzített, összetett feladatot kell elvégeznie. Ebbe beletartozik a forrás és célállomások szimulálása, az implementációk konfigurálása, a kom-

3. ábra Tesztelrendezés a ROHC protokoll automatikus együttműködés-teszteléshez



presszor és dekompresszor közti csatorna monitorozása, valamint a forrás által kiküldött és a célállomás által fogadott csomagok összehasonlítása is.

A tesztrendszer az alábbi komponensekből áll:

- **F** – forrásként viselkedik, előállítja a kompresszor számára az IP csomagokat.
- **C** – célállomást helyettesíti, fogadja a dekompresszortól érkező csomagokat.
- **K** – e komponensek felelnek a kompresszor és a dekompresszor helyes beállításáért (konfigurálás).
- **M** – monitorozás a feladata, figyeli a kompresszor és a dekompresszor közötti ROHC üzeneteket.
- **P** – protokoll üzeneteket képes kiküldeni a csatornára, így lehetővé válik a helytelen viselkedés vizsgálata is. Ha a kompresszor és a dekompresszor nem áll egymással közvetlen kapcsolatban, hanem ezen a komponensen keresztül kötjük őket össze, akkor bizonyos csomagok eldobásával és helyettük hibás csomagok küldésével még a negatív viselkedést is tesztelhetjük.

Egy automatizált együttműködés teszteset végrehajtása több lépésből áll. Először a kompresszort és a dekompresszort kell megfelelően konfigurálni. Mivel erre nincs egységes interfész, ezért olyan megoldásra van szükség, ami minden rendszeren elérhető. Egyik lehetséges megoldás egy telnet kapcsolat létrehozása a tesztrendszer és a tesztelendő implementációk között. Így egy parancssoros felület érhető el, amin keresztül az esetek nagy részében a konfigurálás elvégezhető. Az egyes parancsok természetesen továbbra is implementáció-függőek lesznek, de a parancsokat paraméterként is kezelhetjük, ezzel a tesztsorozat már implementáció-függetlenné tehető.

A következő lépésben megvizsgáljuk, hogy egy adott bemenő IP csomag-sorozatra a tesztelt rendszer megfelelő választ ad-e. A szimulált forrás előálltja az előre definiált csomagokat és elküldi azokat a kompresszornak. A kompresszor előállítja a tömörített ROHC csomagokat, majd ezekből a dekompresszor visszaállítja az IP csomagokat. Az ellenőrzés kétféle módon történik, egyrészt a tesztrendszer figyeli a kompresszor és a dekompresszor közötti forgalmat, másrészt pedig összehasonlítja a forrás által kiküldött és a célállomás által fogadott IP csomagokat. Így nem csak azt tudjuk megállapítani, hogy a tömörítés és visszaállítás sikeres volte, hanem azt is, hogy a kompresszor megfelelő hatékonysággal működik-e.

Ez a tesztelrendezés egyébként nem csak tisztán együttműködés-tesztelésre használható. A monitorozó komponens megfelelő programozásával egyfajta konformancia jellegű együttműködés-tesztek végrehajtására is alkalmas. Vagyis akkor is lehet sikertelen egy ilyen teszt, ha az eredeti IP csomagok visszaállítása sikeres volt (vagyis az együttműködés sikeres), de a kompresszor nem a megfelelő vagy nem a szabvány által előírt legnagyobb hatékonyságot biztosító ROHC csomagformátumot választotta (vagyis a szabvánnyal nem konform).

5. Összefoglalás

A cikkben bemutattuk a konformancia-tesztelés és az együttműködés-tesztelés menetét és tulajdonságait. Látható, hogy egyik módszer sem jobb a másiknál, egyszerűen csak különbözőek. Sem a konformancia-, sem az együttműködés-tesztekkel nem lehet 100% biztonsággal ellenőrizni egy protokoll-implementációt. Ennek szemléltetéséhez nézzük meg az alábbi két példát:

1. példa:

Egy specifikáció szerint egy adott üzenetre 1 másodpercen belül kell az implementációnak választ küldenie. Az implementáció választ is küld, amely szintaktikailag és szemantikailag is helyes, azonban 2 másodperces késéssel. Ebben az esetben a konformancia-teszt sikertelen, de az együttműködés teszt – a másik implementáció viselkedésétől függően – még lehet sikeres.

2. példa:

Egy protokoll implementáció képes kapcsolatot felépíteni a szabványban meghatározott módon, így a konformancia teszt sikeres. Viszont nem képes ezt olyan ütemben vagy mennyiségben megtenni, ahogyan azt egy másik implementáció várná, így az együttműködés teszt sikertelen lesz.

Könnyen belátható tehát, hogy az egyik fajta teszt sikeressége nem garantálja a másik fajta teszt sikerét is, hiszen a két fajta teszt célja eltérő; az implementáció más-más tulajdonságát vizsgálja. A gyakorlatban ezért szükség van mindkét tesztelési eljárásra.

Manapság már mindenki kezdi elfogadni, hogy sem a szabványnak való megfelelést igazoló tanúsítvány (konformancia-teszt), sem pedig az implementáció/berendezés együttműködési képességeit vizsgáló teszt önmagában nem elég. Éppen ezért már nem az a kérdés, hogy melyik fajta tesztre van szükség, hanem sokkal inkább az, hogy melyik módszert mikor célszerű használni, milyen sorrendben, és hogyan lehetne a kettőt minél kevesebb ráfordítással végrehajtani.

Irodalom

[1] C. Bormann (ed.):
Robust Header Compression (ROHC),
RFC 3095, 2001 július

[2] ETSI TS 102 237-1 V4.1.1 (2003-12),
Telecommunications and Internet Protocol
Harmonization Over Networks (TIPHON) Release 4:
Interoperability test methods and approaches,
Part 1: Generic approach to interoperability testing