HÁLÓZATBIZTONSÁG FORMÁLIS SZEMMEL

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg (a támogatás száma TÁMOP 4.2.1./B-09/1/KMR-2010-0003).

Dávid Ákos

Pannon Egyetem, Műszaki Informatikai Kar

Csatári János

Szegedi Tudományegyetem, Természettudományi és Informatikai Kar

Beleznay Péter

Fast Lane Kft.

Motiváció

- Napjainkban számítógépes hálózatok hálózzák be a mindennapjainkat.
- Ezzel a biztonság szerepe is felértékelődik, hisz életünk java része már online zajlik, egy kevésbé védett hálózat pedig értékes vadászterületet kínál az azonosságlopáshoz például.
- Szerencsére egyre nagyobb hangsúlyt kap a megfelelő hálózatbiztonság kialakítása kis- és nagyvállalatoknál egyaránt.
- Ennek eredményeképpen használunk tűzfalat, demilitarizált zónát (DMZ), behatolás-érzékelő (IDS) vagy -megelőző (IPS) rendszereket.
- Ha ezeket megfelelően konfiguráljuk, máris biztonságban vagyunk.

Motiváció (folyt.)

- Vagy mégsem? Mi a helyzet a hálózat azon komponenseivel, amelyekben vakon megbízunk: irányító-, szállítási és hitelesítési protokollokkal többek közt?
- A támadások ugyanis egyre alacsonyabb rétegeket céloznak meg, ahol csökken a felhasználói interakció szükségessége.
- Joggal merülhet fel a kérdés, hogy formálisan is meggyőződhetünk-e egy adott protokoll megfelelőségéről? Segíthet-e egy esetleges ellenpélda a protokoll továbbfejlesztésében? Ezekre a kérdésekre keressük a választ.

Bevezetés

- Jelenleg is óriási mennyiségű adat halad az információs szupersztrádán, és ez a mennyiség a videokommunikáció folyamatos előretörésével exponenciálisan emelkedik. 2015-re várhatóan a hálózati forgalom 91%-át videoanyagok továbbítása teszi ki, továbbá kb. 30 millió munkavállaló legalább heti egy napot otthonról fog dolgozni.
- A hálózati infrastruktúra általánosságban készen áll a megnövekedett adatforgalom kiszolgálására.

Bevezetés (folyt.)

- Egyre nagyobb hangsúlyt kap a megfelelő hálózatbiztonság kialakítása kis- és nagyvállalatoknál egyaránt.
- Ennek eredményeképpen használunk jogosultságkezelést, tűzfalat, demilitarizált zónát (DMZ), behatolás-érzékelő (IDS) vagy -megelőző (IPS) rendszereket.

Bevezetés (folyt.)

- A számítógépes bűnözés fejlődésével viszont a hálózat elleni támadások is egyre kifinomultabbá váltak.
- Már nem feltétlenül igényelnek felhasználói interakciót, inkább a hálózat alsóbb rétegeit célozzák. Az itt működő irányító- és szállítási protokollok felelnek az összes áthaladó adat megfelelő irányba történő, legjobb szándékú továbbításáért.
- Ha egy támadás sikerrel jár, akkor az összes áthaladó információ eltéríthető.

Protokollspecifikáció

- A hálózati protokollok a nagyméretű és komplex szoftverrendszerek sajátosságait hordozzák, ezért egymásra épülő, hierarchikus rétegekbe szervezik őket (pl.: ISO/OSI referenciamodell).
- A rétegelt szerkezet lehetővé teszi, hogy az egyes kommunikációs funkciókat egymástól függetlenül definiáljuk, ezzel együtt ezek fejlesztése is külön – eltérő szabványok szerint – történjen.
- Így tulajdonképpen a komponens-elvűség (CBSD) valósul meg.

Protokollspecifikáció (folyt.)

- Egy adott réteghez tartozó protokoll egyfajta fekete dobozként képzelhető el, amely úgy kínál szolgáltatásokat a felette lévő rétegnek, hogy a vele egy szinten lévő egyedekkel kommunikálva igénybe veszi az alatta lévő réteg szolgáltatásait.
- A tényleges protokoll az adott réteg kommunikációs szabályaiból és a felette lévő rétegnek nyújtott szolgáltatásokból (szolgáltatásspecifikáció) tevődik össze.
- A protokoll egyedeinek leírását, valamint a szolgáltatásspecifikációt együttesen protokollspecifikációnak nevezzük.

Modellek

- Állapot-átmeneti modell – A hálózati protokoll olyan esemény alapú egyed, amely üzenetváltásokkal dolgozik. Az adott egyed az olyan tevékenységek (állapotátmenetek) alapján írható le, amelyekkel a külső vagy belső eseményekre reagál. Ezzel készíthető a legegyszerűbb, ám sok általános tulajdonság vizsgálatát lehetővé tevő modell. Túlságosan komplex protokolloknál azonban felléphet az állapotrobbanás problémája.
- Programozási nyelv alapú modell – Egy hálózati protokoll olyan algoritmikus folyamatokat hajt végre, amelyek leírhatók egy magas szintű programozási nyelv elemeivel. Az ilyen modelleket általában nehéz a megfelelő működés szempontjából vizsgálni.

Modellek (folyt.)

- Hibrid modell – Az állapot-átmeneti és a programozási nyelv alapú modell képességeit kombinálja a protokoll leírásához. Az állapot-átmeneti rendszert változókkal és programrészletekkel egészíti ki a nagyobb kifejezőerő érdekében, amely így kiterjesztett véges állapotú géppé válik.

Protokollellenőrzés

- Legyen *P* a szolgáltatásspecifikáció által sugallt eseménysorozatok halmaza, *Q* pedig a szolgáltatási interfészen (a felette lévő réteg felé) megjelenő eseménysorozatok halmaza, amelyet a protokoll egyed-egyed műveletei generálnak. Ez alapján a protokollellenőrzés általánosságban az alábbiak bizonyítását jelenti:
  - *Q Í P*, ahol például a szolgáltatási interfészen előforduló szolgáltatási primitívek minden létező végrehajtását lehetővé teszi a szolgáltatásspecifikáció, valamint
  - *P Í Q*, ahol például a szolgáltatásspecifikációnak eleget tevő szolgáltatási primitívek minden létező végrehajtása megvalósítható a protokoll egyed-egyed műveleteivel.

Protokollellenőrzés (folyt.)

- Q származtatásával a protokoll egyed-egyed műveletei vizsgálhatók, többek között olyan általános tulajdonságok ellenőrizhetők, mint például a teljesség (a protokoll minden egyes rendszerállapotban elfogadja az összes lehetséges bemenő adatot), holtpontmentesség (a protokoll soha nem kerülhet a végállapoton kívül olyan rendszerállapotba, ahonnan nincs továbblépés).
- Véges futású protokollnál az, hogy egy kiindulási állapotból mindig elérhető egy végállapot, nem véges futású protokollnál pedig ellenőrizhető a ciklikus viselkedés, vagyis minden rendszerállapotból van továbblépés.

Formális verifikációs módszerek

- A protokollellenőrzési módszerek általában kétféle megközelítésmódot használnak: szintézist és analízist. Szintézist jellemzően akkor használunk, ha a protokollt olyan tervezési szabályok alkalmazásával hozzuk létre (az informális specifikációból), amelyek garantálják, hogy az eredményként kapott protokoll rendelkezni fog bizonyos tulajdonságokkal. A kívánt tulajdonságok megléte beépül a tervezési szabályokba.
- Analízist akkor használunk, ha a hálózati protokoll specifikációja adott, és a protokoll elemzésével kell igazolni bizonyos elvárt tulajdonságok meglétét. Az állapottér vizsgálata (modellellenőrzés), valamint a programok helyességbizonyítása egyaránt ide tartozik.

Az irányítóprotokollok tulajdonságai

- Lényegében korlátlan számú, több példányban futó, egyszerű, konkurens folyamatról van szó.
- Dinamikus összeköttetéseket feltételezünk, és elvárjuk a hibatűrést.
- A folyamatok szerény komplexitású, diszkrét interfésszel rendelkező, reaktív rendszerek.
- A valós idejűség alapvető fontosságú, mivel bizonyos tevékenységeknek van elévülési idejük, vagy épp arra kell reagálniuk.

Példa: OSPF

- Széles körben használt belső irányítóprotokoll (IGP).
- 2-es verzióját (IPv4) a 2328-as RFC definiálja.
- Két állapotgép van definiálva benne, az interfész FSM (Section 9) és a szomszédsági FSM (Section 10).
- Az RFC tartalmazza az állapotgép leírását, azaz az
  - állapotokat (state),
  - eseményeket (event),
  - átmeneteket (transition),
  - interfészekhez pedig a lehetséges típusokat.
- Modellezésre és ellenőrzésre az NuSMV-t használjuk.

OSPF interfész FSM
interfész-típusok

- Point-to-point
- Broadcast
- NBMA
- Point-to-MultiPoint
- Virtual link

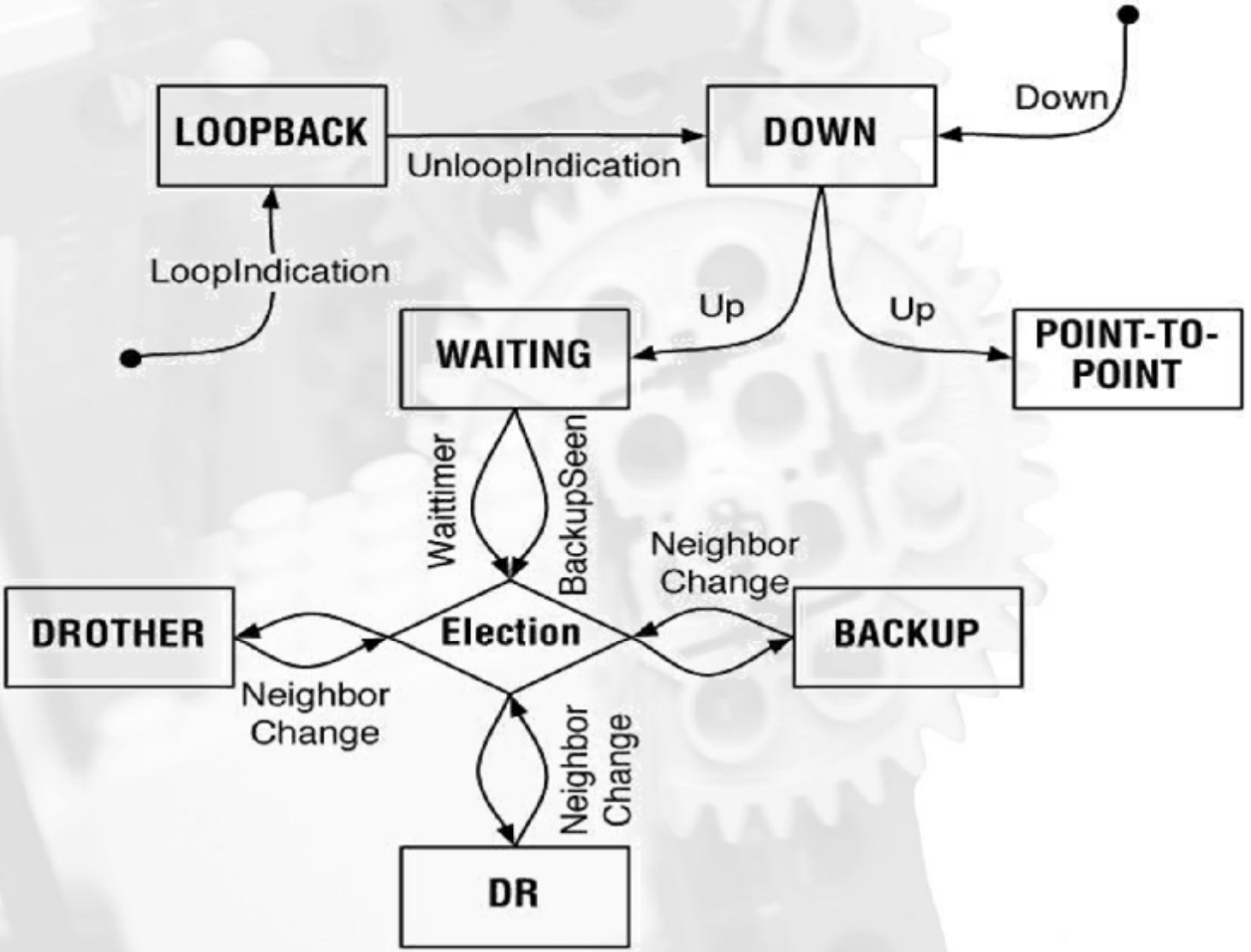
OSPF interfész FSM állapotok

- Down: kezdeti állapot, semmilyen protokollforgalom nem kerül továbbításra.
- Loopback: az interfész normál forgalmat nem továbbít, csak információszerzésre szolgál.
- Point-to-point: az interfész működőképes, és fizikai pont-pont vagy virtual-link kapcsolatot hoz létre.
- Waiting: a forgalomirányító megpróbálja meghatározni a hálózat DR és BDR eszközeit a Hello csomagok figyelésével.
- DR Other: szórásos vagy NBMA hálózatra csatlakozik az interfész, és nem került kiválasztásra, mint DR vagy BDR. Szomszédsági viszonyt hoz létre a DR és BDR forgalomirányítókkal.
- Backup: BDR-nek választották ki a hálózaton az interfészt. Szomszédsági viszonyba lép a hálózaton lévő összes többi eszközzel.
- DR: Kijelölt forgalomirányítónak (Designated Router) választották, és hasonlóan a Backup (tartalék) állapothoz, szomszédsági viszonyba lép az összes többi eszközzel.

OSPF interfész FSM események

- InterfaceUp: a hálózati interfész működőképes.
- WaitTimer: a várakozási időzítő elindul, ennek a végén kerül kiválasztásra a DR és a BDR.
- BackupSeen: a Hello-üzenetek segítségével felfedezett egy BDR-t a hálózaton az interfész.
- NeighborChange: változás történt az interfészhez rendelt szomszédok viszonyában. A DR-t és BDR-t újra kell számolni.
- LoopInd: jelzés, hogy az interfész loopback állapotba vált.
- UnloopInd: jelzés, hogy az interfész kilép a loopback állapotból.
- InterfaceDown: az interfész nem képes tovább működni az alacsonyabb szintű protokollok jelzése alapján.

OSPF interfész FSM



OSPF interfész FSM vizsgálata

- !EF((interface.type = point-to-point xor interface.type = point-to-multipoint xor interface.type = virtual_link) & (interface.state = dr xor interface.state = drother xor interface.state = backup));
- -- specification !(EF (((interface.type = point-to-point xor interface.type = point-to-multipoint) xor interface.type = virtual_link) & ((interface.state = dr xor interface.state = drother) xor interface.state = backup))) is true
- (EF *p* is true in a state s_0 if *there exists* a series of transitions $s_0 \rightarrow s_1$, $s_1 \rightarrow s_2$, ..., $s_{n-1} \rightarrow s_n$ such that *p* is true in s_n.)

OSPF interfész FSM vizsgálata

- !EF((interface.type = point-to-point xor interface.type = point-to-multipoint xor interface.type = virtual_link) & (interface.state = point-to-point));
- -- specification !(EF (((interface.type = point-to-point xor interface.type = point-to-multipoint) xor interface.type = virtual_link) & interface.state = point-to-point)) is false

Ellenpélda

Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
interface.type = point-to-multipoint
interface.state = down
interface.event = interface_up
neighbor.state = down
neighbor.event = hello_received
-> Input: 1.2 <-
_process_selector_ = interface
running = FALSE
neighbor.running = FALSE
interface.running = TRUE
-> State: 1.2 <-
interface.state = point-to-point

OSPF interfész FSM vizsgálata

- !EF((interface.type = broadcast xor interface.type = nbma) & (interface.state = point-to-point));
- -- specification !(EF ((interface.type = broadcast xor interface.type = nbma) & interface.state = point-to-point)) is true

OSPF interfész FSM vizsgálata

- !EF((interface.type = broadcast xor interface.type = nbma) & (interface.state = dr xor interface.state = drother xor interface.state = backup));
- -- specification !(EF ((interface.type = broadcast xor interface.type = nbma) & ((interface.state = dr xor interface.state = drother) xor interface.state = backup))) is false

Ellenpélda

Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 2.1 <-
interface.type = broadcast
interface.state = down
interface.event = interface_up
neighbor.state = down
neighbor.event = hello_received
-> Input: 2.2 <-
_process_selector_ = interface
running = FALSE
neighbor.running = FALSE
interface.running = TRUE
-> State: 2.2 <-
interface.state = waiting
-> Input: 2.3 <-
-> State: 2.3 <-
interface.event = wait_timer
-> Input: 2.4 <-
-> State: 2.4 <-
interface.state = backup

OSPF szomszédsági FSM állapotok

- Down: kezdeti állapot, nincs előzetes információ a szomszédról.
- Attempt: (csak NBMA-hálózatok esetén) nem érkezett információ a szomszédoktól, ezért direkt kapcsolatfelvételre van szükség.
- Init: Hello-csomag érkezett a szomszédtól, de kétirányú kommunikációra még nem került sor (azaz a forgalomirányító még nem jelent meg a szomszéd Hello-üzenetében).
- 2-Way: ebben az állapotban a kapcsolat már kétirányú.
- ExStart: ez az állapot az első lépés a szomszédsági viszony létrehozásában. Ebben a lépésben kerül meghatározásra a „mester” forgalomirányító a kezdeti DD (Database Description) sorszám alapján.
- Exchange: a forgalomirányító a teljes LS adatbázisát átküldi DD csomagok segítségével a szomszédjának.
- Loading: ebben az állapotban a forgalomirányító LSR-kéréseket küld a szomszédjának az előző állapotban felfedezett, de még meg nem kapott legújabb LSA-hirdetésekért.
- Full: ez az állapot jelenti a teljes szomszédsági viszonyt.

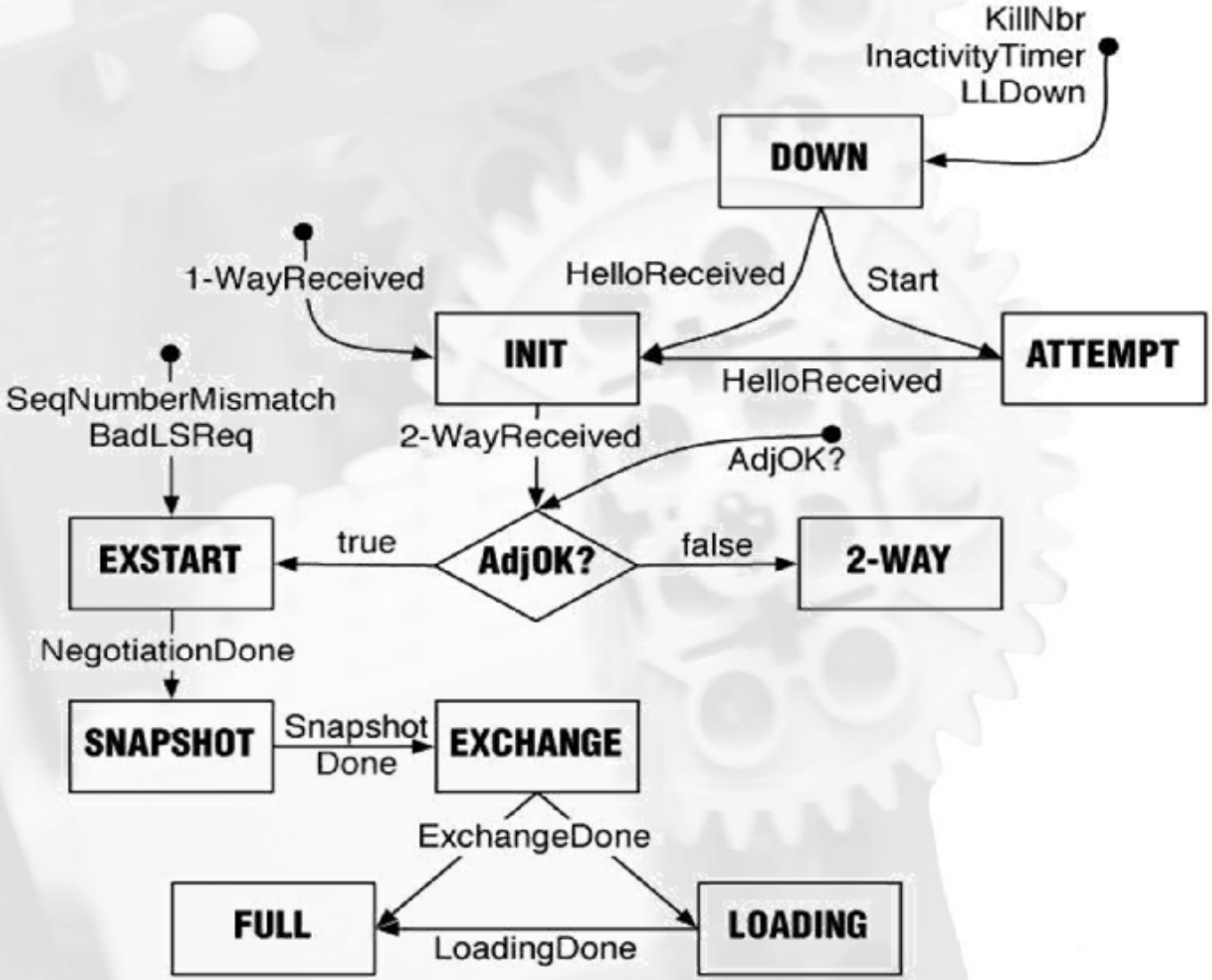
OSPF szomszédsági FSM események

- HelloReceived: egy Hello-csomag érkezett a szomszédtól.
- Start: ettől az állapottól kezdve HelloInterval időközönként Hello-csomagokat küld a forgalomirányító a szomszédnak.
- 2-WayReceived: kétirányú kapcsolat épült fel egy szomszédos forgalomirányítóval.
- NegotiationDone: a mester/szolga viszony kialakult és a DD sorszámok kicserélése megtörtént a szomszédos forgalomirányítóval.
- ExchangeDone: mindkét forgalomirányító eljuttatta a másikhoz az összes DD-csomagját.
- LoadingDone: megérkezett az összes LS-frissítés a szomszédtól.
- BadLSReq: az adatbázisban nem található LSA-hoz érkezett LS-kérés, ez hibát jelöl az adatbázis-szinkronizáció során.
- AdjOk?: el kell döntenie a forgalomirányítónak, hogy szükséges-e létrehoznia/fenntartania a szomszédsági viszonyt.

OSPF szomszédsági FSM események

- SeqNumberMismatch: egy DD-csomag érkezett váratlan DD-sorszámmal, beállított Init-bittel, vagy az Options mező különbözik a legutóbb kapott DD-csomagétól. Ez szintén hibát jelöl a szomszédsági viszony létrehozásában.
- 1-Way: egy olyan Hello-csomag érkezett a szomszédtól, amiben nincs megjelölve a forgalomirányító, azaz (még) nincs kétirányú kapcsolat.
- KillNbr: jelzi, hogy a kommunikáció lehetetlen a szomszéddal, és visszaáll Down állapotba.
- InactivityTimer: az Inactivity Timer elindítása, azaz egy ideje már nem érkezett Hello-csomag a szomszédtól, és visszaáll Down állapotra.
- LLDown: alsóbb szintű protokollok jelzik a szomszéd elérhetetlenségét.

OSPF szomszédsági FSM



OSPF szomszédsági FSM vizsgálata

- !EF(neighbor.state = full);
- -- specification !(EF neighbor.state = full) is false

Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 3.1 <-
interface.event = interface_up
interface.state = down
interface.type = point-to-point
neighbor.state = down
neighbor.event = hello_received
-> Input: 3.2 <-
_process_selector_ = neighbor
running = FALSE
neighbor.running = TRUE
interface.running = FALSE

OSPF szomszédsági FSM vizsgálata

-> State: 3.2 <-
neighbor.state = init_state
-> Input: 3.3 <-
-> State: 3.3 <-
neighbor.event = two-way_received
-> Input: 3.4 <-
-> State: 3.4 <-
neighbor.state = exstart
neighbor.event = hello_received
-> Input: 3.5 <-
-> State: 3.5 <-
neighbor.event = negotiation_done
-> Input: 3.6 <-
-> State: 3.6 <-
neighbor.state = exchange
neighbor.event = adj_ok_question
-> Input: 3.7 <-
-> State: 3.7 <-
neighbor.event = exchange_done
-> Input: 3.8 <-
-> State: 3.8 <-
neighbor.state = full
neighbor.event = bad_lsr_req

Két OSPF-router DR/BDR választása

- Szimuláció: két forgalomirányító egy-egy broadcast típusú interfésszel kapcsolódik egymáshoz eltérő prioritással. Hogyan változnak az állapotok?
- r1_priority := 1; r2_priority := 2;
- !EF(r1.int1.state = dr & r2.int1.state = backup);
- -- specification !(EF (r1.int1.state = dr & r2.int1.state = backup)) is false

Két OSPF-router DR/BDR választása

Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
r1.int1.event = interface_up
r1.int1.state = down
r1.int1.type = broadcast
r2.int1.event = interface_up
r2.int1.state = down
r2.int1.type = broadcast
r2_priority = 2
r1_priority = 1
-> Input: 1.2 <-
_process_selector_ = r1.int1
running = FALSE
r2.running = FALSE
r2.int1.running = FALSE
r1.running = FALSE
r1.int1.running = TRUE
-> State: 1.2 <-
r1.int1.state = waiting
-> Input: 1.3 <-
-> State: 1.3 <-
r1.int1.event = wait_timer
-> Input: 1.4 <-
-> State: 1.4 <-
r1.int1.state = election (könnyítés)
-> Input: 1.5 <-
-> State: 1.5 <-
r1.int1.state = dr

Két OSPF-router DR/BDR választása

-> Input: 1.6 <-
_process_selector_ = r2.int1
r2.int1.running = TRUE
r1.int1.running = FALSE
-> State: 1.6 <-
r2.int1.state = waiting
-> Input: 1.7 <-
-> State: 1.7 <-
r2.int1.event = wait_timer
-> Input: 1.8 <-
-> State: 1.8 <-
r2.int1.state = election
-> Input: 1.9 <-
-> State: 1.9 <-
r2.int1.state = backup

Konklúzió

- A valódi működést helyesen szimuláló modellek kerültek kifejlesztésre.
- Ezeket később tovább lehet bővíteni:
  - több-forgalomirányítós, több-interfészes modellek
  - LSA-hirdetések szinkronizációjának folyamata
  - hitelesítés vizsgálata

Irodalomjegyzék

1. RFC 2328: OSPF Version 2 (1998) available at http://www.ietf.org/rfc/rfc2328.txt
2. Bhargavan, K., Obradovic, D., Gunter, C. A. (2002) Formal Verification of Standards for Distance Vector Routing Protocols. Journal of the ACM (JACM) , Volume 49, Issue 4, pp. 538-576.
3. Karlin, J., Forrest, S., Rexford, J. (2008) Autonomous security for autonomous systems. Computer Networks, Oct. 2008.
4. Kent, S., Lynn, C., Seo, K. (2000) Secure border gateway protocol (S-BGP). J. Selected Areas in Communications, vol. 18, pp. 582-592.
5. Manna, Z., Pnueli, A. (1995) Temporal Verification of Reactive Systems-Safety, Springer.
6. Lécz, B., Zömbik, L. (2004) Hálózati protokollok biztonsági tesztelése. Híradástechnika, 2004/3: 2-6.
7. McDaniel, P., Aiello, W., Butler, K., Ioannidis, J. (2006) Origin authentication in interdomain routing. Computer Networks, Nov. 2006.
8. Sidhu, D., Chung, A., Blumer, T. P. (1991) Experience with formal methods in protocol development. ACM SIGCOMM Computer Communication Review, Volume 21, Issue 2, pp. 81-101.