Teljesítményvizsgálat elosztott tesztkomponensekkel

CSORBA J. MÁTÉ, PALUGYAI SÁNDOR

Ericsson Magyarország, Test Competence Center
{mate.csorba, sandor.palugyai}@ericsson.com

Lektorált

Kulcsszavak: TTCN-3, teljesítményvizsgálat, párhuzamos tesztkomponensek

Cikkünkben TTCN-3 tesztkörnyezeten alapuló teljesítmény- és terhelésvizsgálatokra alkalmas szoftverkomponensek elemzésével foglalkozunk. Az általunk adott módszer célja teljesítménytesztek fejlesztésének támogatása a tesztkomponensek teljesítmény-kritikus viselkedésének előrejelzésével, már a korai tervezési fázis során. Módszerünk legfőképpen a tesztkomponensek megvalósításában rejlő sorbanállási mechanizmusokat veszi figyelembe, más TTCN-3 specifikus tulajdonságok mellett, melyek befolyásolhatják egy tesztrendszer késleltetéseit. A módszer használatával megbecsülhetjük azt a forgalmi korlátot mely alatt nagy biztonsággal érdemes TTCN-3 alapú tesztkomponenseket használnunk az adott vizsgálatainkra.

1. Bevezetés

Távközlési berendezések, hardver és szoftver eszközök tesztelése kapcsán vizsgálatok széles skálájáról beszélhetünk. Végezhetünk funkcionális teszteket, protokollmegfelelőségi vizsgálatokat (konformancia-tesztelés), vizsgálhatjuk távközlési berendezések, valamint szoftver-megvalósítások együttműködési képességét, robusztusságát (stabilitás és megbízhatóság vizsgálata). Távközlési szoftverek fejlesztése során nagy jelentősége van az úgynevezett regressziós teszteknek, amelyek a folyamatosan fejlesztett szoftver újabb és újabb verzióinak gyakran ismételt vizsgálatára alkalmasak.

Általában egy rendszer reakciós idejét, illetve terhelhetőségét teljesítménytesztekkel állapíthatjuk meg. A teljesítményvizsgálatok körében is többfajta vizsgálati céllal találkozhatunk, úgy mint skálázhatóság vizsgálata, mennyire könnyen bővíthető egy rendszer. Stressz-teszt, azaz a rendszer viselkedése hirtelen és/vagy rendkívül túlméretezett terhelés alatt [1]. Teljesítményvizsgálat során azt a várható környezetet és terhelést próbáljuk meg előállítani, amellyel majd a működő rendszernek szembe kell néznie, és arra keressük a választ, mekkora az a terhelés, amelyet a rendszer még elvisel, illetve mekkora hibaarányt produkál, késleltetése hogyan változik a terheltség függvényében. A legtöbb esetben több felhasználó párhuzamos, egyidejű működésének szimulációjával vizsgáljuk az adott távközlési rendszert és miközben változó számú felhasználói populációt szimulálunk, vizsgáljuk a rendszerrel történő kommunikációt. Ez a tesztek részéről hatékony és elosztott működést feltételez.

Mivel a vizsgált megvalósítást fekete dobozként kezeljük, és pusztán kívülről stimuláljuk protokoll-üzenetek formájában, a tesztrendszernek képesnek kell lennie a megfelelő mennyiségű tesztüzenet előállítására. Ez gyakran a teszteket futtató hardver platform képességeibe ütközhet, amit általában még több hardverelem munkába állításával tudunk megelőzni. Ugyanakkor a felhasználók nem szekvenciális, egyidejű viselkedését úgyszintén több, párhuzamosan futó processz alkalmazásával tudjuk hatékonyan szimulálni.

Akár egy elég erős hardveren futó, több párhuzamos felhasználót szimuláló, akár a rendelkezésre álló több gépet hatékonyan kihasználó tesztek esetében elosztott tesztkomponensekről beszélhetünk. Mindkét esetben felmerül a kérdés, milyen stratégia szerint osszuk szét a funkciókat a tesztkomponensek között, valamint miként helyezzük el az egyes komponenseket a vizsgálatokban résztvevő hardveren, munkaállomásokon. A megfelelő stratégiának, mely szerint a tesztkomponenseket elhelyezzük, figyelembe kell vennie a munkaállomások kiépítettségét és ebből adódó eltérő sebességét csakúgy, mint az egy adott munkaállomáshoz rendelt szimulált felhasználók számát.

Egy végpont a tesztek futtatásához, elegendő felhasználó imitálásához, nem mindig rendelkezik elég erőforrással. További erőforrás szükségletet jelent, hogy az egyes tesztkomponensek egymás között is kommunikálhatnak és a konkrét teszt megvalósításától függően kommunikálnak is. Például koordinációs, szinkronizációs, start/stop üzenetek cseréje, futás közbeni statisztikai és kiértékelést segítő lekérdezések. A komponensek viselkedésének vizsgálatánál ezt a belső kommunikációt is figyelembe kell venni.

A teszteléshez használt szoftverkomponensek elemzésének célja, hogy támogassa a munkaállomásokon történő szétosztásukat. Az elosztási mechanizmusokat megkülönböztetjük aszerint, hogy a mechanizmus statikus, vagy dinamikus működésű, tehát csak a teszt futtatását megelőzően, vagy futtatás közben is használható [2].

A különböző elosztási stratégiák mindegyike figyelembe veszi a kialakított hardver/szoftver struktúra valamely tulajdonságát a döntések megkönnyítésére, úgy mint a processzorterhelést, memóriafoglalást, I/O műveletek gyakoriságát, a hálózati interfészek paramétereit. A felsorolt tulajdonságok figyelembevétele történ-

het statikusan, a komponensek elosztását megelőző számítások támogatásához, illetve folyamatosan egy kialakított hardver-monitorozó keretrendszer segítségével.

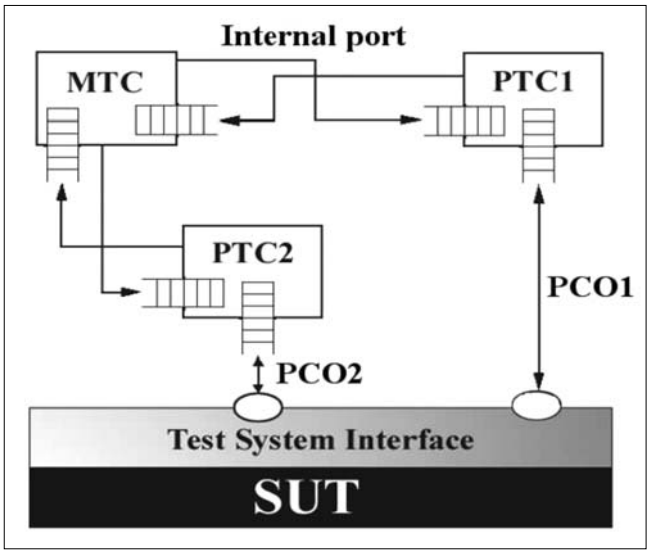
A következőkben egy sztochasztikus modellt adunk teljesítményvizsgáló komponensek viselkedésének becslésére [3], mellyel statikus elosztás valósítható meg. Használatával egy adott tesztkomponens üzenetvesztési valószínűségeit becsülhetjük meg különböző bemenő paraméterek mellett, és választ kaphatunk arra, hogy a komponens a bemenő paraméterekben leírt hardveren képes lesz-e a teszt-specifikációban leírt követelmények teljesítésére.

2. Teljesítménytesztelés elosztott TTCN-3 tesztkomponensekkel

Vizsgálataink során a Testing and Test Control Notation version 3 (TTCN-3) [4] tesztnyelvre, és az ezen a nyelven történő fejlesztésre és elosztott tesztvégrehajtásra koncentrálunk. A TTCN-3 tesztkörnyezetet széles körben alkalmazzák távközlési rendszerek tesztjeinek megvalósítására, de egyéb területeken is megvetette már a lábát. Jó példa erre gépjárművek kommunikációs rendszereinek vizsgálata.

Az utóbbi időben egyre nagyobb az igény a TTCN-3 nyelv használatára teljesítményvizsgálatok körében, ennek megfelelően több tanulmány is foglalkozik a nyelv ilyen irányú lehetőségeivel, kiterjesztésével és alkalmazásával [5-8]. Mivel a TTCN-3 nyelv egy magas szintű specifikációs nyelv, kérdéses lehet a hatékonysága egy alacsonyabb szintű, hardverközeli megvalósítással szemben, teljesítménytesztek írása esetén. Azonban számos úttörő projekt, számítás és mérés megmutatta, hogy a magas szintű tesztnyelv képes a hardver által nyújtott korlátok és erőforrások teljes kihasználására abban az esetben, ha a tesztek írása körültekintően és néhány ökölszabály betartásával történik.

1. ábra
TTCN-3 tesztkomponensek és kommunikációs portok egy lehetséges elrendezése



A TTCN-3 nyelvű párhuzamos tesztkomponensek (Parallel Test Components, PTCs) statikus vagy dinamikus elosztására kétfajta megközelítés ismert. A komponenseket elosztó mechanizmus megvalósítható egy köztes réteg (middleware) beiktatásával és az elosztó mechanizmus külön implementálásával, valamilyen más nyelven [9]. Ez azonban, a middleware megvalósításától függően, egy újabb szűk keresztmetszet lehet a tesztrendszer számára. Ezért mi a komponensek elosztását és az úgynevezett terheléselosztást (load control) szintén TTCN-3 nyelven valósítjuk meg, kihasználva a nyelv által adott lehetőségeket. A nyelv által biztosított platformfüggetlenségnek köszönhetően a PTC-k elosztása könnyen testre szabható különböző hardver környezet használatához.

3. Tesztkomponensek sztochasztikus vizsgálata

Az általunk felállított sztochasztikus modell TTCN-3 nyelvű PTC-k eseményfeldolgozó kapacitásának figyelembevételével képes megbecsülni az adott komponens üzenetvesztési valószínűségét, azaz használhatóságának korlátját. A tesztrendszer üzenetei lehetnek belső (koordinációs) üzenetek, illetve a külvilággal történő kommunikációt megvalósító, tényleges tesztüzenetek. Az elosztott tesztrendszer több PTC-ből és egy fő, vezérlő komponensből (Main Test Component, MTC) állhat. A köztük lévő kommunikáció az úgynevezett test portokon (Points of Control and Observation, PCOs) keresztül történik, csakúgy mint a vizsgálat célját képező rendszerhez (System Under Test, SUT) történő kapcsolódás *(1. ábra)*.

A kommunikációs portok elméletileg egy-egy különálló várakozó sorral rendelkeznek, amely a gyakorlatban a tesztkomponenst futtató munkaállomás memóriájának egy szelete. Ezek a PCO-k egymással versengenek a rendszer erőforrásaiért. A versenyhelyzetet meghatározza az egyes portokra érkező igények érkezési intenzitása, illetve a tesztkomponens által futtatott teszteset jellemzői, bonyolultsága.

A komponensek vizsgálatára egy diszkrét idejű kvázi születési-halálozási folyamat (D-QBD) alapú modellt állítunk fel [10], melynek segítségével a komponenshez tartozó PCO-k versenyhelyzetének elemzésével becslést adunk a komponens üzenetvesztésére vonatkozóan. A becslés történhet a tényleges megvalósítást megelőzően (visszacsatolás a tervezési fázisban), illetve a megvalósítást követően a PTC-k munkaállomásokra történő elosztási stratégiájának támogatására (statikus elosztás).

A PTC viselkedését leíró modell a következő TTCN-3 specifikus mechanizmusokat veszi figyelembe: a komponensen futó teszteset alternatíváinak – melyek leírják a lehetséges végrehajtási utakat (a SUT-tel történő kommunikáció protokolljának véges állapotú automatája alapján) – működése. Az úgynevezett snapshot logika [11] – mely a TTCN-3 üzenetfeldolgozás során hasz-

nált mintaillesztési mechanizmusa – által okozott késleltetés. Valamint, a bejövő és kimenő üzenetek teszt portokon (PCO-kon) kialakuló sorbanállásának hatásai.

Ahhoz, hogy ezeket a mechanizmusokat a modell képes legyen számításba venni, a következő bemenő paraméterekkel rendelkezik:

(a) illeszkedési valószínűségek (találati arányok) a tesztkomponens alternatíváinak minden egyes (a TTCN-3 kódban található *alt* struktúra összes) bejegyzésére,

(b) érkezési intenzitások a tesztkomponens minden egyes portjához külön-külön.

Minden egyes PCO-hoz a modell feltételezése szerint Poisson-eloszlás alapján érkeznek a protokoll-üzenetek, így ebben az esetben az eloszlás várható értékét adjuk meg. Továbbá bemenő paraméter még – mivel diszkrét idejű modellről van szó –, a diszkrét időegység, melyet annak az időnek a függvényében állítunk be, ami a komponenst futtató munkaállomáson szükséges egy beérkező üzenet megvizsgálásához (template matching művelet végrehajtásához). Ez a paraméter gyakorlatilag a CPU sebességétől függ. Ezeknek a paramétereknek a segítségével építjük fel a D-QBD-t egyértelműen definiáló állapot-átmeneti mátrixot (*P* mátrix).

Esetünkben a D-QBD folyamat egy speciális QBD lesz, mely irreguláris 0. szinttel rendelkezik és a szintek száma véges, ennek megfelelően alakul a folyamatot leíró *P* mátrix és almátrixai:

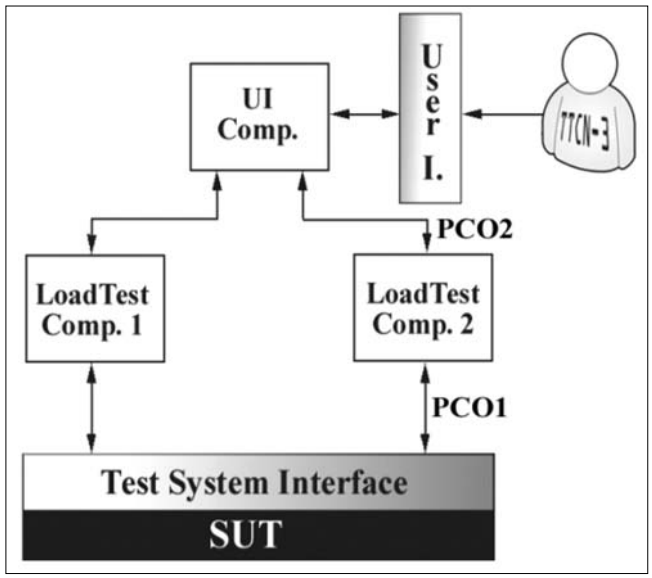
$$\underline{\underline{P}} = \begin{bmatrix} \underline{\underline{B}}^* & \underline{\underline{C}}^* & 0 & 0 & 0 & \dots \\ \underline{\underline{A}}^* & \underline{\underline{B}} & \underline{\underline{C}} & 0 & & \\ 0 & \underline{\underline{A}} & \underline{\underline{B}} & \underline{\underline{C}} & 0 & \\ 0 & 0 & \underline{\underline{A}} & \underline{\underline{B}} & \underline{\underline{C}} & \\ & & & \dots & \dots & \underline{\underline{C}} \\ & & & & \underline{\underline{A}} & \underline{\underline{B}}^{**} \end{bmatrix} \quad (1)$$

Az így kapott QBD modell ezután hatékonyan számítható mátrix-analitikus módszerekkel, melynek során a folyamat állandósult állapotbeli megoldását keressük, és így kapjuk a folyamat tetszőleges állapotának előfordulási valószínűségét [12]. Az általunk definiált véges, kétdimenziós modell két konkurens PCO-t használó tesztkomponens leírására alkalmas. Kettőnél több PCO leírása N-dimenziós modell felállítását teszi szükségessé, melynek számítása jelenleg is izgalmas feladat, de nem megoldhatatlan [13].

4. Alkalmazási példa

Tételezzük fel, hogy tesztrendszerünk három párhuzamos TTCN-3 komponensből áll *(2. ábra)*. A rendszer felhasználója egy grafikus felületen keresztül kapcsolódik egy komponenshez, ami a felületet kezeli, és vezérli a két másik, ténylegesen teljesítményvizsgálatot végző PTC-t.

Ilyen elrendezésben vizsgáljuk a *LoadTestComp.2* nevű PTC működését, és kiszámítjuk a csomagvesztés valószínűségét a komponens belső megvalósításának

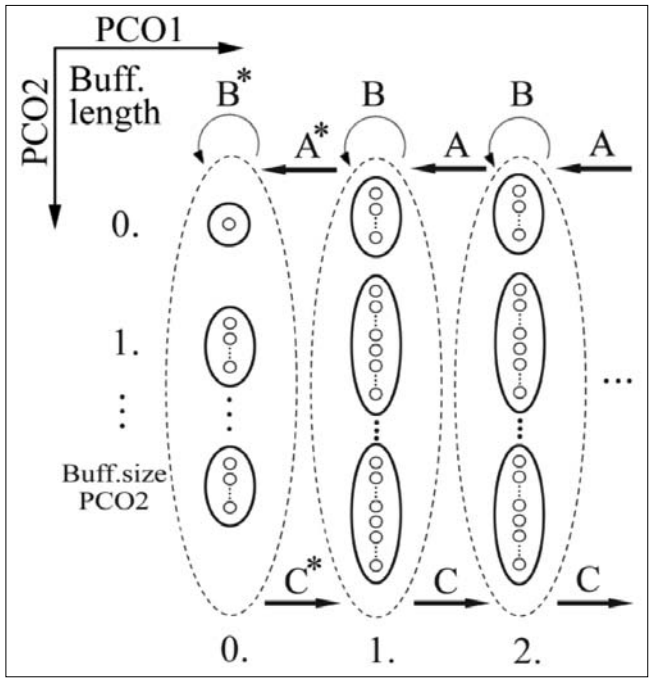


2. ábra Tesztelrendezés három PTC-vel

és két kommunikációs portjára (PCO1 és PCO2) érkező csomagok intenzitásának függvényében. Az így kapott kétdimenziós modell látható a *3. ábrán,* a teszt-portok pufferelését leíró szintekkel és – a szinteken belül – a TTCN-3 komponens alternatíváit leíró fázisokkal.

A vizsgálat tárgyául (SUT) ebben az esetben sokféle távközlési berendezés elképzelhető, például egy ATM kapcsoló, vagy egy IP útvonalválasztó. A példabeli elrendezésben a vizsgált komponens a PCO2 jelű porton keresztül kommunikál a felhasználóval kapcsolatot tartó *UI* komponenssel, míg a tényleges teljesítményvizsgálat protokoll üzenetei a PCO1 porton érkeznek és távoznak. Ez egy teljesítmény vizsgáló komponens esetében azt is jelenti, hogy a két port forgalmának intenzitása jelentősen eltér. A felhasználó konfiguráló üze-

3. ábra
A kétportos PTC-t modellező kétdimenziós QBD folyamat



netei, illetve lekérdezései (PCO2) – már csak az emberi reakcióidő viszonylagos lassúsága miatt is – rendkívül ritkának tekinthetők a SUT-tel történő kommunikációhoz hasonlítva, melynek nagyságrendje másodpercenként több ezer üzenetet is jelenthet.

A tesztkomponensek tervezéséhez megvizsgálhatjuk a PTC által még éppen kezelhető forgalom intenzitását a modell segítségével. Tekintsük bemenő paraméternek az érkezési intenzitásokat és számítsuk ki a modell állandósult állapotbeli megoldását. Így kapjuk a stacionárius állapot-valószínűségeket (π_i vektorok) a modell minden állapotára és minden szintjére vonatkozóan (2). A vektorok kiszámítása minden *n*-re a rekurzív *R* mátrix segítségével történhet, mely az állapot-átmeneti mátrix almátrixainak *(A, B, C)* segítségével kapható [12].

$$\underline{\pi}_1 = \underline{\pi}_0 \cdot \underline{\underline{R}}; \underline{\pi}_2 = \underline{\pi}_1 \cdot \underline{\underline{R}}; \ldots \underline{\pi}_n = \underline{\pi}_0 \cdot \underline{\underline{R}}^n \qquad (2)$$

Amint rendelkezünk az állandósult állapotbeli megoldással, kiszámíthatjuk a terhelés okozta versenyhelyzetben alulmaradó porton létrejövő üzenetvesztés valószínűségét (3). Ehhez összegeznünk kell azon állapotok valószínűségét, amelyekben tartózkodva a rendszer már nem képes újabb érkezéseket kezelni. Ezek az állapotok az irreguláris 0. szinten és a reguláris szinteken ($j = 1 \ldots \infty$) a PCO2 pufferméretének megfelelő utolsó (vertikális, lásd 3. ábra) szintek állapotai (*i, k* állapotok).

$$\Pr(Loss_{PCO2}) = \sum_i \pi_{0i} + \sum_{j=1}^{\infty}\left(\sum_k \pi_{j_k}\right) = \ldots \underline{\pi}_{0_l} + \sum_{j=1}^{\infty}(\underline{\pi}_{j_m}) =$$

$$\underline{\pi}_{0_l} + \sum_{j=1}^{\infty}(\underline{\pi}_{1_m} \cdot \underline{\underline{R}}^{j-1}) = \underline{\pi}_{0_l} + \underline{\pi}_{1_m} \cdot \sum_{k=0}^{\infty}(\underline{\underline{R}}^k) =$$

$$= \underline{\pi}_{0_l} + \underline{\pi}_{1_m} \cdot (\underline{\underline{I}} - \underline{\underline{R}})^{-1} \qquad (3)$$

Az egyszerűsítések után kapott formulával ezután megbecsülhetjük, mekkora az az érkezési intenzitás, a *LoadTestComp.2* névre hallgató komponens esetében, melyet adott üzenetvesztési valószínűség alatt még kezelni tud. Ezáltal a tesztek adott forgalomra méretezhetővé válnak még a tényleges futtatást megelőzően.

5. Összefoglalás

Teljesítménytesztek tervezése és megvalósítása bonyolult feladat TTCN-3-mal, de bármilyen más eszközzel is. Körültekintő tervezésnek kell megelőznie a teljesítménytesztek megvalósítását, hogy hatékonyan működjenek a végrehajtásra szolgáló platformon. A létrehozott tesztkomponens modellünk lehetővé teszi meglévő tesztek elemzését is, valamint képes egyfajta visszacsatolást nyújtani a tesztek tervezői számára.

További terveink közt szerepel a model továbbfejlesztése, hogy alkalmassá váljon több konkurens PCO-t használó PTC-k leírására is egy N-dimenziós modell megalkotásával.

Irodalom

[1] R. Binder, Testing Object-Oriented Systems: Models, Patterns and Tools. Addison-Wesley, 2000.

[2] G. Din, S. Tolea, I. Schieferdecker, „Distributed Load Tests with TTCN-3”, TestCom 2006, LNCS 3964, pp.177–196.

[3] M. J. Csorba, S. Palugyai, S. Dibuz, Gy. Csopaki, „Performance Analysis of Concurrent PCOs in TTCN-3”, TestCom 2006, LNCS 3964, pp.149–160.

[4] ETSI ES 201 873-1 (V3.1.1) Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language, 2005.

[5] Z. Wang, J. Wu, X. Yin, X. Shi, B. Tian, „Using TimedTTCN-3 in Interoperability Testing for Real-Time Communication Systems”, TestCom 2006, LNCS 3964, pp.324–340.

[6] H. Neukirchen, Z. Ru Dai, J. Grabowski, „Communication Patterns for Expressing Real-Time Requirements Using MSC and their Application to Testing”, TestCom 2004, LNCS 2978, pp.144–159.

[7] Z. Ru Dai, J. Grabowski, H. Neukirchen, „Timed TTCN-3 Based Graphical Real-Time Test Specification”, TestCom 2003, LNCS 2644, pp.110–127.

[8] Z. Ru Dai, J. Grabowski, H. Neukirchen, „Timed TTCN-3 – A Real-time Extension for TTCN-3”, TestCom 2002, pp.407–424.

[9] I. Schieferdecker, T. Vassiliou-Gioles, „Realizing Distributed TTCN-3 Test Systems with TCI”, TestCom 2003, LNCS 2644, pp.110–127.

[10] G. Latouche, V. Ramaswami, „Introduction to Matrix Analytic Methods in Stochastic Modeling”, The American Statistical Association and the Society for Industrial and Applied Mathematics, 1999. pp.83–99., pp.221–237.

[11] ETSI ES 201 873-4 (V3.1.1) – Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 4: TTCN-3 Operational Semantics, 2005.

[12] M. F. Neuts, „Matrix-Geometric Solutions in Stochastic Models”, Johns Hopkins University Press, 1981. pp.81–107., pp.112–114.

[13] T. Osogami, „Analysis of multi-server systems via dimensionality reduction of Markov chains”, PhD. thesis, Computer Science Department, Carnegie Mellon University, 2005.