

# 國立中山大學九十四學年度博士班招生考試試題

科目：資訊管理論文評述(一)【資管系選考】

共 | 頁 第 | 頁

1. 請仔細閱讀下面這篇研討會論文並且 (50%)
  - (1) 評述這篇論文整篇研究有那些主要的缺點與問題？
  - (2) 如果要進行一篇相類似的研究，你的研究主題、理論、模式、假說、分析方法的設計會如何？

## 企業文化、組織結構與引進 Intranet 意願之研究

### 摘要

Intranet 係以瀏覽器為介面的 Web 技術，提供一個企業高效率、低成本的溝通工具，許多理論認為資訊科技的引進，與企業文化、組織結構兩者存在密切的關聯性，

為瞭解國內企業對引進 Intranet 意願的考量因素，本研究以 Davis [1989]提出的科技接受模式 (Technology Acceptance Model; TAM) 為基礎，配合「企業文化」與「組織結構」變數，針對國內企業資訊部門主管引進 Intranet 的認知、意願及使用行為，與企業文化、組織結構因素之間的關係進行問卷調查。問卷對象根據中華民國資訊經理人協會之成員名冊為主，加上民國 86 年天下雜誌一千大企業名冊隨機取樣，共寄發 600 份問卷，回收 112 份，回收率 18.7%，研究結果發現：

- 一、目前國內實際使用及規劃 Intranet 的企業佔 71%
- 二、資訊主管對於 Intranet 的認知、態度與意願，影響企業的實際引進行為。
- 三、Intranet 容易轉換與使用的科技特性，使得資訊主管對 Intranet 的易用認知直接影響引進意願。
- 四、企業文化沒有影響資訊主管的認知，但分權式組織結構負影響資訊主管的有用認知。
- 五、企業文化、結構正式化與企業引進 Intranet 的意願與實際行為有關。

此外，實證結果發現目前國內資訊主管引進 Intranet 的因素，仍是由本身專業認知所主導，缺乏以企業特性著眼的考量。

關鍵字：企業文化、組織結構、Intranet 企業網路、科技接受模型。

## 壹、研究背景與目的

Intranet 從 1995 年中期從美國出現至今，在各企業已然蔚為風潮，以國內目前的環境而言，新科技(如 Intranet)的引進、測試、發展、推廣等工作多為資訊部門主管負責。因此，資訊部門對於 Intranet 的測試或使用上的認知(perception)、態度(attitude)、意願(intention)，對於 Intranet 的引進便格外重要，尤其在 Intranet 尚未完全成熟之際，資訊部門主管更是扮演舉足輕重的角色。

但是，Intranet 的科技特性對組織結構與企業文化均能產生極大的影響。所以，科技引進不單純是資訊部門的事，企業文化、組織結構皆會影響組織接受此一新科技的過程。國內許多科技引進失敗的例子，可能常導因於資訊主管只專注於新技術的特性與趨勢，而忽略了文化與組織的情況。

因此，瞭解企業的組織特性是否影響資訊主管引進 Intranet 的認知與意願，對於 Intranet 能否與企業特性結合而產生效益，扮演重要的角色；此外，探討資訊主管的認知與實際引進 Intranet 的關係，也有助於瞭解新科技的引進過程，這是本研究的動機所在。

## 貳、文獻探討

### 一、企業文化與資訊科技的探討部分

Orlikowski [1992]綜合過去組織與科技關係學者的研究，分析這些研究結果，提出科技結構化模式(Structurational Model of Technology)她認為科技與人際作用、企業制度特性互有關聯性，依據她的推論，科技、組織、變革模式(人際行為)，三者有緊密互動的關係，企業在引進群組軟體的科技時，必需配合其變革模式與組織構面的相關要素。

許多學者認為企業文化應視為策略規劃(strategic planning)所應考慮的一項要素。並強調企業策略與企業文化的相容性(compatibility)。意即企業策略的選擇，必須要配合企業文化 [Davis,1984; Scholz,1987]

Morton [1991]提出 90 年代的平衡管理模型(Management of the 1990s Equilibrium Model)描述策略、科技、個人角色及企業文化、管理程序、組織結構等五項要素需要平衡。

Schein [1992]認為文化是一種群體現象，文化的形式取決於溝通的形式，意即藉由組織成員的溝通與互動，才能創造及維持企業的文化。

Orlikowski 與 Hofman[1997]在對群組軟體(groupware)的變革行為研究中，她認為科技、組織、變革模式(人際行為)，三者有緊密互動的關係，企業在引進群組軟體的科技時，必需配合其變革模式與組織構面的相關要素。同時，在其個案研究的企業裡，發現團隊導向(team-oriented)的企業文化較能適應新的群組軟體技術。

### 二、組織結構與資訊科技的探討

Thompson [1967]根據部門之間的依賴性(inter-dependence)及協調性(coordination)對其科技進行分類，他定義出三種型態的科技，並建議最好以三種不同的協調方式來管理科技。Robbins [1990]整理學者對組織結構與科技之間的關係，依組織結構的分類認為慣例性科技明顯的有較低的複雜度；而組織正式化與慣例性工作有關連；最後是當組織規則與條例最少時(即正式化程度越小)，慣例性科技與集權控制就會有關聯性。

Grover & Goslar [1993]在探討美國企業對傳真等十五項通訊科技的應用時，認為影響企業採用通訊科技的因素可分為環境因素、組織因素、以及資訊系統因素。其研究結果發現，兩個有顯著影響的因素是環境複雜度、及決策制定的集權程度。

李經遠[1990]調查台灣中小型製造業，發現企業其科技變動與組織構面的變化兩者有明確的關係；三凱 [1997]發現對 Internet 工具認知重要的企業具有較高的正式化程度，也會呈現出較長的 Internet 使用歷史。

綜合上述學者研究，顯示出組織結構與企業資訊科技的使用之間有密切的關連性存在；而近幾年來亦有不少學者進行組織結構與資訊科技配合的相關研究，更加突顯此一議題，日益受到重視的程度。

### 三、科技接受認知行為的相關研究部分

Davis[1989]提出 TAM 用來分析使用者對科技的認知程度與意圖，進而預測實際使用科技的行為。

TAM 提出之後，相關的研究論文陸續發表。Davis[1989]曾就 IBM 實驗室 120 使用者進行電子郵件及文書軟體的使用調查，研究資訊接受模型的認知易用及認知有用。Igbria [1997]則針對中小企業進行個人電腦的接受情形調查，並以 TAM 為基礎，探討使用者接受個人電腦的認知行為。Szajna[1996]調查 61 位商學研究所學生使用電子郵件以實證評估技術接受模型。許多研究加強驗證的結果，Chau[1996]認為 TAM 是所有決定使用資訊系統/資訊科技(IS/IT)的研究中，最有影響力的研究模型之一。

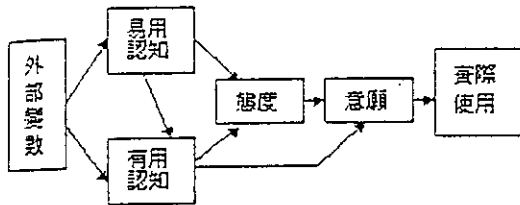


圖 1 TAM 模型 [Davis,1989]

參、研究架構與假設

一、研究架構

本研究根據 Fulk[1987]所提出的社會資訊處理模式(Social Information Processing Model)認為組織採用溝通科技的原因，在於其組織對於該媒體特性的認知，並以 Davis[1989]的 TAM 模型為基礎，配合科技結構化模型[Orlikowski,1992] 等理論的支持，將「企業文化」與「組織結構」等組織變數與企業引進 Intranet 的認知與意願進行相關的實證研究。研究架構如下圖所示：

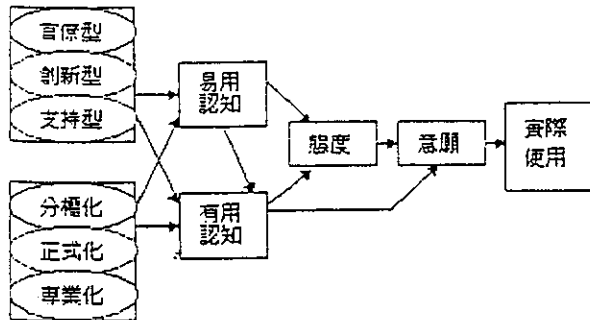


圖 2 研究架構

二、研究變數

(一)、企業文化：採用 Wallach[1983]，將企業文化區分為三類：「官僚型」、「創新型」、「支持型」。這種企業文化之分類，已廣被許多研究者採用。

1. 官僚型 (bureaucratic) 文化：

有清楚的責任及授權，工作為系統化及固定化，其組織通常為穩定、小心、及成熟的。

2. 創新型 (innovative) 文化：

此類文化的環境通常變化較為激烈及動態性。工作通常較具創造性，並且充滿了風險。

3. 支持型 (supportive) 文化：

是一種重開放、和諧的工作環境，組織中具有高度的支持、鼓勵開放的工作環境。

(二)、組織結構：張緯良[1980]綜合學者對組織結構的分類，將組織結構化定義為組織中相關的政策與活動，而這些政策與活動引導或限制了組織成員的行為。包括：

1. 集權化：組織權力分配在垂直軸上的情形。

2. 正式化：企業中各種作業程序與方法、規章制度書面化的程度。

3. 專業化：組織中不同的職稱數來定義，顯示其分工精細的程度。

(三)、主管對引進 Intranet 的認知與態度：

有用認知：相信使用系統能增加工作效率。

易用認知：相信使用系統能輕鬆地達成目的。

態度：對 Intranet 技術的正面或負面感覺。

(四)、意願：依照 Fishbein[1975]的定義，研究之意願，指個人對於採用 Intranet 資訊架構所能接受及使用的動機程度。

(五)、企業目前實際採用 Intranet 的情況：分正式使用、規劃、觀察中、不使用四類情況。

肆、研究方法

一、研究對象與資料蒐集

本研究係屬實證性研究(empirical study)，旨在驗證理論架構及假設。具體言之，係以國內企業資訊部門主管為研究母體。

研究的基本樣本框架(sampling frame)一為中華民國資訊經理人協會的成員名冊，該會成員資格必須具有任職資訊部門主管兩年以上的經歷，故其樣本框架充分具有母體所代表的特性，樣本框架二為天下雜誌一千六企業，由於技術上無法掌握各企業之資訊主管姓名資料，故在郵寄名條上，皆註明為「資訊部經理」為收件人，以降低非資訊主管作答的情況出現。

樣本的選擇上，以中華民國資訊經理人協會為主要取樣樣本，樣本數 340 份，再加上 96 年天下雜誌一千六企業的調查結果為抽樣名冊，採用「等距取樣法」，抽取 260 家企業，共發出問卷 600 份，做為本研究的樣本對象。

本研究採郵寄問卷法，自民國八十六年十一月十六日寄發，至八十七年一月十日截止回收，共計發出 600 份問卷，回收有效問卷 112 份，有效回收率達 18.7%。

一、信度與效度檢定

本研究對於二個項目以上問題的變數進行 Cronbach's  $\alpha$  係數的計算，下表所示為本研究各構念之信度係數，本研究之所有構面的信度相關係數皆大於 0.6 之可接受範圍。

本研究之問卷內容(衡量變數所有層面的項目)係以理論為基礎，參考以往學者類似研究之問卷內容加以修訂，並與實務或學術專家討論及進行前測(pretest)，因此應具有相當的內容效度(content validity)。

研究並利用效標關連效度(criteria-related validity)來測量效度係數，以檢定內在效度指標

(intrinsic validity index)，測量結果如下表 1 所示，效度皆達 0.76 以上，可謂研究的統計效度良好。

表 1：問卷各構面信度及效度

變數	信度 (Cronbach's $\alpha$ )	效標效度 (Max V)
官僚型	0.558	0.811
創新型	0.849	0.921
支持型	0.849	0.921
分權化	0.632	0.795
正式化	0.830	0.911
專業化	0.721	0.849
有用程度	0.926	0.962
易用程度	0.902	0.949
引進態度	0.927	0.963
引進意願	0.919	0.959

伍、研究結果

一、相關分析

我們將主要研究變數進行相關分析，列出 Pearson 相關係數矩陣如下：

表 2：Pearson 積差相關係數矩陣

變項	1	2	3	4	5	6	7	8	9	10
1.官僚型	—									
2.創新型	.43**	—								
3.支持型	.335**	.740**	—							
4.分權化	.126	.492**	.544**	—						
5.正式化	.542**	.478**	.430**	.310**	—					
6.專業化	.172	.114	-.54	.138	-.004	—				
7.易用認知	.192*	.187*	.131	.147*	.14	.135	—			
8.有用認知	.200*	.135	.087	-.094	.212*	-.086	.292**	—		
9.態度	.207	.04	.083	.084	.116	-.064	.373**	.669**	—	
10.意願	.274**	.253**	.301**	.204*	.333**	-.016	.369**	.471**	.603**	—
11.實際使用	.215*	.227*	.261**	.182	.214*	-.051	.261**	.269**	.396**	.469**

\*\* 0.01 顯著水準 \*\*\* 0.005 顯著水準

將上表之相關係數矩陣為基礎，簡化以下圖表示(粗線表示 0.01 顯著水準)可以看出三種企業文化型態與引進意願及使用情形之間的相關性存在。

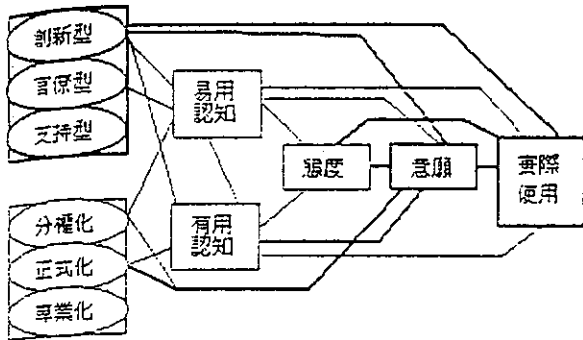


圖3 研究變數關係圖

二、整體架構驗證

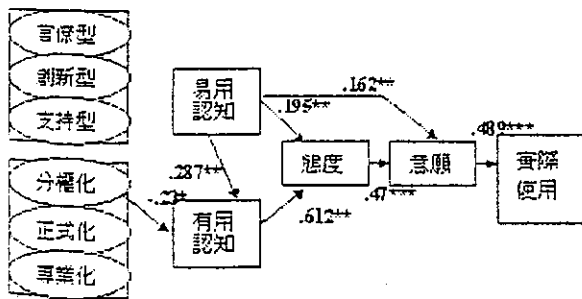
進一步對各依變項進行複迴歸分析，變異解釋能力  $R^2$  值列表如下：

自變項	依變項	$R^2$
文化、結構、易用認知	有用認知	0.18***
易用認知、有用認知	態度	0.48***
易用認知、有用認知、態度	意願	0.40***
易用認知、有用認知、態度、意願	實際使用	0.24***

\*\*\*0.005 顯著水準

表3 複迴歸分析

並與圖2之整體研究架構進行驗證，估計其路徑係數，以瞭解自變項對依變項的作用力大小。



\* 0.05 \*\*0.01 \*\*\*0.005 顯著水準

圖4 路徑圖

陸、結論與建議

一、企業對於引進 Intranet 的態度、意願及實際使用情形。

本研究利用李克特五點尺度量表進行調查，發現資訊主管對於 Intranet 的引進意願較高，樣本總平均數 4.02，顯示多數企業資訊主管對於採用 Intranet 科技所能接受或使用的動機程度甚高。在態度方面，總平均數 4.38 大於其 Intranet 的意願程度；顯示多數資訊主管個人對於 Intranet 抱持正面及肯定的態度。

依據使用情況進行分類，其中已經使用與規劃使用兩者佔樣本總數 71%，顯示 Intranet 的科技熱潮在國內已成氣候，受到企業的重視。

二、資訊主管對 Intranet 的認知影響引進意願。

1、資訊主管對 Intranet 的認知與態度，影響企業引進 Intranet 的意願與行為。

雖然資訊科技、資訊系統(IT/IS)的引進，與企業內外諸因素有關，但本研究發現資訊主管的 Intranet 易用/有用認知直接影響其引進態度，而態度亦影響引進意願及使用行為。換句話說，資訊主管認為 Intranet 對於企業是有幫助的(有用的或容易使用的)，就能推動企業引進及使用 Intranet。

因此，在實務上，如欲加強推動企業引進 Intranet，則加強資訊主管本身對於 Intranet 之利益認知，確有其必要性。

2、有用認知沒有直接影響意願，但易用認知直接影響引進意願。

依研究之路徑圖可以看出，有用認知的影響並不直接，而是經過態度影響其意願。而易用認知直接影響意願，推論起因於 Intranet 系統容易轉換與容易使用的科技特性。同時，這也是本研究與 Davis 的 TAM 模型最大不同處，科技容易使用的特性，會直接影響使用者對該科技的使用意願。換句話說，在網路的環境中，讓使用者相信網路產品是容易使用的，比讓他相信該產品的有用性更為重要。

三、企業特性影響對 Intranet 認知與意願。

1、企業文化沒有直接影響 Intranet 的認知程度；推

論由於 Intranet 軟硬體成本低廉，讓資訊主管在考慮 Intranet 時，未能受到企業文化特性的影響。

- 2、分權化結構負影響 Intranet 的有用認知程度；資訊科技的引進對於集權化與分權化組織均有所貢獻，但分權化結構負影響其有用認知，推論係結構上的集權化造成組織在溝通上的困難，及管理上的困擾所致。
- 3、資訊主管引進 Intranet 的認知，主要由其專業知識主導，並未慎重考慮組織因素(文化、結構)的影響層面；正如 Orlikowski[1994]在 Notes 企業網路的研究中，發現技術人員及資訊主管僅著重於使用 Notes 來達成其技術目標，但卻未由企業效益、顧客服務等組織觀點來考量新科技。然而，Intranet 的引進對於未來企業文化與結構型態上都有產生變革的可能，但資訊主管卻未能顧及這些因素，因此國內企業引進 Intranet 是否能發揮溝通、協調與合作，進而創新的科技特性，確是值得企業高階主管與資訊主管應該留意的地方。

#### 四、組織構面與 Intranet 意願、使用之關係。

- 1、企業文化與引進 Intranet 之意願及行為有關；整體的企業文化力量，仍與 Intranet 的使用有關，而在實務應用上，當企業的整體文化無法配合企業引進 Intranet 的策略時，由於資訊主管仍是企業內部資訊系統規劃、建置的重要人物。因此增強資訊主管對 Intranet 利益認知的方式，亦可促使企業引進 Intranet 之意願與行為。
- 2、組織結構正式化與企業引進 Intranet 意願及實際使用有關；由於 Intranet 電子公文文化的特性，將使組織正式化運作更有效率，所以正式化的組織較有意願使用 Intranet。

#### 柒、未來研究與限制

##### 一、研究限制

Schein[1992]認為由問卷調查法來推測企業文化，測得的部份有可能是組織氣氛或規範。他建議應加入客觀的第三者或實地觀察企業。基於此點，使我們

們在推論文化與其他變數的關係上，解釋能力受到部分限制。

本研究以問卷調查法為主，受訪者在填答過程中可能因誤解題意而造成偏差。此外，研究之部份量表係由國外論期刊翻譯而來，設計上未能考量本土化企業情況，這些構成研究上的限制。

##### 二、後續研究建議

Fishbein 的 TRA 理性行為理論及 Davis 的 TAM 理論，都是認為人們通常表現出和行為意向一致的行為。這些理論都假定人類的思考和行為是理性的，但它忽略了影響行為的非理性因素，這類的因素包括有限的的能力或資源、對行為的主觀控制感及外在的限制與機會。正因如此，在意願與使用行為之間，也可能有企業資源、機會等中間變數之不同而存在變因。

未來研究者可針對此點加以研究，以擴大研究範圍。

#### 參 考 文 獻

- 三凱、范錚強、李延平“網際網路認知與應用—以台灣地區五個產業為例”第三屆國際資訊管理研究暨實務研討會論文集 1997, pp.175-182
- 李經遠“企業組織結構與科技、規模間關係之研究,”  
交大管理學報, 第十卷第一期, 1990
- 張緯良“資訊技術之應用對組織的影響”政大企研所, 1980.
- Chau, P.Y.K., "An Empirical Assessment of a Modified Technology Acceptance Model, *Journal of Management Information System*", 1996, (13:1) pp. 185-201.
- Davis, F.D., "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology", *MIS Quarterly*, 1989(13:3) pp. 319-340
- Davis, S.M., *Managing Corporate Culture*, Cambridge, 1984.
- Fishbein, M. & Ajzen, I. "Belief, Attitude, Intention and Behavior: An Introduction to Theory and Research." Addison-Wesley, Reading, 1975

- Fulk, J., Steinfield, C. W., Schmitz, J., & Power, J. G. "A Social Information Processing Model of Media use in Organizations." *Communication Research*, 14, pp.529-552
- Grover, V. and Goslar, M., "The Initiation, Adoption and Implementation of Telecommunication Technologies in U.S. Organizations," *Journal of Management Information Systems*, (10:1), Summer 1993, pp. 141-163.
- Morton, M.S. "The Corporation of the 1990s" Oxford University Press, 1991
- Orlikowki, W. J., "The Duality of Technology: Rethinking The Concept of Technology in Organizations" *Organization Science* (3:3) August 1992 pp.398-427.
- Orlikowki, W.J., and Gash, D.C., "Technological Frames: Making Sense of Information Technology in Organizations," *ACM Transactions on Information System*, (12:12) April 1994, pp.174-207.
- Orlikowski, W.J., and Hofman, J.D., "An Improvisational Model for Change Management: The Case of Groupware Technologies," *Sloan Management Review*, Winter 1997, pp.11-21.
- Robbins S. P., "Organization Theory: Structure, Design, and Applications" 1990.
- Schein, E.H., "Organizational Culture and Leadership" 2<sup>nd</sup>, San Francisco, CA: Jossey-Bass, 1992
- Scholz, C., "Corporate Culture and Strategy: The Problem of Strategic Fit," *Long Range Planning*, (20:4), 1987, pp. 78-87.
- Szajna, B., "Empirical Evaluation of the Revised Technology Acceptance Model." *Management Science*, (42:1), 1996, pp.87-91.
- Thompson, J., "Organizations in Action", 1967
- Wallach, E.J., "Individuals and Organizations : The Culture Match." *Training and Development Journal*, February 1983 pp.29-36.



# 國立中山大學九十四學年度博士班招生考試試題

科目：資訊管理論文評述(二)【資管系選考】

共 3 頁 第 1 頁

1. The area of information management has evolved in the past forty years from a new area of computer applications to become an essential resource for strategic competition. In particular, the wide adoption of the Internet has dramatically changed the way information is processed and utilized. Please answer the following three questions:
- (1) The scope and role of information management in a modern enterprise (15%),
  - (2) How are the major information technologies and particularly the Internet affecting the development of information systems (15%)
  - (3) Please read the article below ("Drive Traffic to Your Website") and identify two potential issues for research in information systems and describe why they are interesting. The issues may be technical or managerial. (20%)

Note: Your answer must be written in English.

## **Drive Traffic to Your Web Site**

Source: <http://www.businesscentre.bell.ca/>

Web sites are not billboards on the highway that can be seen by drivers. They are more like trees in the vast cyberspace forest. That begs the question: how do you drive people to your site?

Traditional marketing - newspapers, magazines, radio and TV advertising - is costly. However, if that's how you promote your company, then include your Web site address in all advertising, as well as on business cards and stationery. Be creative. For instance, if you want to drive university students to your funky site, try on-campus washroom billboards.

### **Broadcast your message**

If you have a customer contact list, consider direct mail. Barbara Sybal, president, GFX Printing Services of Mississauga, Ontario, prints full colour images on postcards. To market her Web site, she sends out promotional postcards with her domain name emblazoned on them.

Fax broadcasts can also be used to drive traffic to Web sites. "We've concluded that the most effective way to bring people to Duncans

MindLeaders is fax broadcast," says John Duncan, manager of Duncans in Regina, Saskatchewan.

Duncan sends faxes during business hours and keeps them brief. "We use little or no graphics, so the fax goes through in under 30 seconds, even on slower fax machines." Of the faxes he sends, 10% result in site hits and he religiously removes names from the fax list upon request.

Then there are electronic means.

#### **Internet marketing tools**

To start with, optimize your Web site for search engines. Meta tags should include key words potential visitors or customers would use when searching for your site, products or services. Don't include inappropriate words like "Pamela Anderson," "Playboy" or even "Disney." This will drive inappropriate traffic to your site and it infringes on trademarks.

It can take time to get your site listed in some search engines. If your audience is national or international, consider paying for listings in popular search engines. "I know the importance of search engines and have paid a few of them their submission fees," says Sybal.

If you find high-traffic sites that reach your target market, consider banner ads. Or use more innovative ways to promote your company. Chris Boothe, president of Mississauga, Ontario-based FavorWare Corporation, ran a software giveaway contest on software developer Web portals to promote his company's business automation solutions. The contest cut through the "bombardment of product pitches" developers receive and drew traffic to the company's Web site.

#### **Low-cost options**

Budget tight, but time on your hands? Engage in guerrilla marketing. Include your Web site address and short description of your business in your e-mail signature, making your site a click away every time you send e-mail.

Participate in discussion groups. Marketing a service for small businesses? Join a small business mailing list. Include your Web address in your

signature, but participate constructively in discussions. If you blatantly promote your site, you will make enemies and find yourself banned from moderated discussion groups.

Reciprocal Web links are another means of site promotion. Find non-competitive sites that attract your target market and link to them. Ask for reciprocal links back. The "Web ring" is a variation on reciprocal links. Each site in the ring of related sites uses a "next" hot-link so visitors can travel from site to site with the click of a mouse button.

#### **Keep them coming back for more**

Once you have people on your site, how do you keep them coming back? Try opt-in e-mail marketing. Most people with an e-mail address would like to strangle spammers, but opt-in e-mail can be used to market Web sites.

An Ipsos-Reid study found 92% of Canadian Internet users are e-mail junkies, and most are receptive to opt-in or permission-based e-mail marketing. Nearly four out of five Canadian Internet users have opted in to e-mail campaigns, however, 77% of them have unsubscribed because information received was not as advertised or was sent too frequently. So, when you lure people in with the promise of informative newsletters, fulfil the promise. But don't bombard them.

To automate the subscribe and unsubscribe process and the delivery of e-mail newsletters, use customer relationship management (CRM) or Web-enabled contact management software. Make it easy for people to unsubscribe from your mailing list so you don't turn potential customers into angry consumers.

#### **Respect the customer**

Bottom line? To get your name, or rather your Web site address, out there, use appropriate electronic and traditional marketing. Be innovative, but respect the fact that not everyone wants to visit your site. The last thing you want are angry consumers using online guerrilla techniques to bad-mouth your company because you got in their face in an inappropriate manner.

# 國立中山大學九十四學年度博士班招生考試試題

科目：資訊科技論文評述(一)【資管系選考】

共 | 頁 第 | 頁

## PhD Entrance Exam -IT

Please read the attached paper and answer the following questions. Note that the time allocated to this sub-test is only 80 minutes, you should budget your time carefully. It is suggested that you first go through the whole paper quickly and pay attention to the Abstract, Introduction and Conclusions. Then, you should carefully read the following five questions and try to find out proper answers from the paper.

1. What is the problem the paper is aiming to solve? 5%
2. Describe how TCP protocol is being used in the proposed solution. 5%
3. Describe how ICMP protocol is being used in the proposed solution. 5%
4. What is the main purpose of using NTP in a computer? 5%
5. How can this technique be applied to the fields of computer security or electronic commerce? 20%
6. What is the originality of the paper? 10%

# Remote physical device fingerprinting

Tadayoshi Kohno  
CSE Department, UC San Diego  
tkohno@cs.ucsd.edu

Andre Broido  
CAIDA, UC San Diego  
broido@caida.org

kc claffy  
CAIDA, UC San Diego  
kc@caida.org

## Abstract

We introduce the area of remote physical device fingerprinting, or fingerprinting a physical device, as opposed to an operating system or class of devices, remotely, and without the fingerprinted device's known cooperation. We accomplish this goal by exploiting small, microscopic deviations in device hardware: clock skews. Our techniques do not require any modification to the fingerprinted devices. Our techniques report consistent measurements when the measurer is thousands of miles, multiple hops, and tens of milliseconds away from the fingerprinted device, and when the fingerprinted device is connected to the Internet from different locations and via different access technologies. Further, one can apply our passive and semi-passive techniques when the fingerprinted device is behind a NAT or firewall, and also when the device's system time is maintained via NTP or SNTP. One can use our techniques to obtain information about whether two devices on the Internet, possibly shifted in time or IP addresses, are actually the same physical device. Example applications include: computer forensics; tracking, with some probability, a physical device as it connects to the Internet from different public access points; counting the number of devices behind a NAT even when the devices use constant or random IP IDs; remotely probing a block of addresses to determine if the addresses correspond to virtual hosts, e.g., as part of a virtual honeynet; and unanonymizing anonymized network traces.

## 1 Introduction

There are now a number of powerful techniques for remote operating system fingerprinting, i.e., techniques for remotely determining the operating systems of devices on the Internet [2, 3, 5, 27]. We push this idea further and introduce the notion of remote physical device fingerprinting, or remotely fingerprinting a physical device, as opposed to an operating system or class of devices, without the fingerprinted device's known cooperation. We accomplish this

goal to varying degrees of precision by exploiting microscopic deviations in device hardware: clock skews.

**CLASSES OF FINGERPRINTING TECHNIQUES.** We consider three main classes of remote physical device fingerprinting techniques: passive, active, and semi-passive. The first two have standard definitions — to apply a passive fingerprinting technique, the fingerprinter (measurer, attacker, adversary) must be able to observe traffic from the device (the fingerprintee) that the attacker wishes to fingerprint, whereas to apply an active fingerprinting technique, the fingerprinter must have the ability to initiate connections to the fingerprintee. Our third class of techniques, which we refer to as semi-passive fingerprinting techniques, assumes that after the fingerprintee initiates a connection, the fingerprinter has the ability to interact with the fingerprintee over that connection; e.g., the fingerprinter is a website with which the device is communicating, or is an ISP in the middle capable of modifying packets en route. Each class of techniques has its own advantages and disadvantages. For example, passive techniques will be completely undetectable to the fingerprinted device, passive and semi-passive techniques can be applied even if the fingerprinted device is behind a NAT or firewall, and semi-passive and active techniques can potentially be applied over longer periods of time; e.g., after a laptop connects to a website and the connection terminates, the website can still continue to run active measurements.

**METHODOLOGY.** For all our methods, we stress that the fingerprinter does not require any modification to or cooperation from the fingerprintee; e.g., we tested our techniques with default Red Hat 9.0, Debian 3.0, FreeBSD 5.2.1, OpenBSD 3.5, OS X 10.3.5 Panther, Windows XP SP2, and Windows for Pocket PC 2002 installations.<sup>1</sup> In Table 1 we summarize our preferred methods for fingerprinting the most popular operating systems.

Our preferred passive and semi-passive techniques exploit the fact that most modern TCP stacks implement the

<sup>1</sup>Our techniques work for the default installs of other versions of these operating systems; here we just mention the most recent stable versions of the operating systems that we analyzed.

Technique and section	Class	NTP	Red Hat 9.0	OS X Panther	Windows XP
TCP timestamps, Section 3	passive	Yes	Yes	Yes	No
TCP timestamps, Section 3	semi-passive	Yes	Yes	Yes	Yes
ICMP tstamp requests, Section 4	active	No	Yes	No	Yes

Table 1. This table summarizes our main clock skew-based physical device fingerprinting techniques. A “Yes” in the NTP column means that one can use the attack regardless of whether the fingerprintee maintains its system time with NTP [19]. One can use passive and semi-passive techniques when the fingerprintee is behind a NAT or current generation firewall.

*TCP timestamps option* from RFC 1323 [13] whereby, for performance purposes, each party in a TCP flow includes information about its perception of time in each outgoing packet. A fingerprinter can use the information contained within the TCP headers to estimate a device’s clock skew and thereby fingerprint a physical device. We stress that one can use our TCP timestamps-based method even when the fingerprintee’s system time is maintained via NTP [19]. While most modern operating systems enable the TCP timestamps option by default, Windows 2000 and XP machines do not. Therefore, we developed a trick, which involves an intentional violation of RFC 1323 on the part of a semi-passive or active adversary, to convince Microsoft Windows 2000 and XP machines to use the TCP timestamps option in Windows-initiated flows. In addition to our TCP timestamps-based approach, we consider passive fingerprinting techniques that exploit the difference in time between how often other periodic activities are supposed to occur and how often they actually occur, and we show how one might use a Fourier transform on packet arrival times to infer a device’s clock skew. Since we believe that our TCP timestamps-based approach is currently our most general passive technique, we focus on the TCP timestamps approach in this paper.

An active adversary could also exploit the ICMP protocol to fingerprint a physical device. Namely, an active adversary could issue ICMP Timestamp Request messages to the fingerprintee and record a trace of the resulting ICMP Timestamp Reply messages. If the fingerprintee does not maintain its system time via NTP or does so only infrequently and if the fingerprintee replies to ICMP Timestamp Requests, then an adversary analyzing the resulting ICMP Timestamp Reply messages will be able to estimate the fingerprintee’s system time clock skew. Default Red Hat 9.0, Debian 3.0, FreeBSD 5.2.1, OpenBSD 3.5, and Windows 2000 and XP and Pocket PC 2002 installations all satisfy the above preconditions.

PARAMETERS OF INVESTIGATION. Toward developing the area of remote physical device fingerprinting via remote clock skew estimation, we must address the following set of interrelated questions:

- (1) For what *operating systems* are our remote clock skew estimation techniques applicable?
- (2) What is the *distribution* of clock skews across multiple fingerprintees? And what is the *resolution* of our clock skew estimation techniques? (I.e., can one expect two machines to have measurably different clock skews?)
- (3) For a single fingerprintee, can one expect the clock skew estimate of that fingerprintee to be relatively *constant* over long periods of time, and through reboots, power cycles, and periods of down time?
- (4) What are the effects of a fingerprintee’s *access technology* (e.g., wireless, wired, dialup, cable modem) on the clock skew estimates for the device?
- (5) How are the clock skew estimates affected by the *distance* between the fingerprinter and the fingerprintee?
- (6) Are the clock skew estimates *independent of the fingerprinter*? I.e., when multiple fingerprinters are measuring a single fingerprintee at the same time, will they all output (approximately) the same skew estimates?
- (7) How much *data* do we need to be able to remotely make accurate clock skew estimates?

Question (6) is applicable because common fingerprinters will probably use NTP-based time synchronization when capturing packets, as opposed to more precise CDMA- or GPS-synchronized timestamps. Answers to the above questions will help determine the efficacy of our physical device fingerprinting techniques.

EXPERIMENTS AND HIGH-LEVEL RESULTS. To understand and refine our techniques, we conducted experiments with machines that we controlled and that ran a variety of operating systems, including popular Linux, BSD, and Microsoft distributions. In all cases we found that we could use at least one of our techniques to estimate clock skews of the machines, and that we required only a small amount of data, though the exact data requirements depended on the operating system in question. For the most popular operating systems, we observed that when the system did not use NTP- or SNTP-based time synchronization, then the TCP timestamps-based and the ICMP-based techniques yielded

approximately the same skew estimates. This result, coupled with details that we describe in the body, motivated us to use the TCP timestamps-based method in most of our experiments. We survey some of our experiments here.

To understand the effects of topology and access technology on our skew estimates, we fixed the location of the fingerprinter and applied our TCP timestamps-based technique to a single laptop in multiple locations, on both North American coasts, from wired, wireless, and dialup locations, and from home, business, and campus environments (Table 3). All clock skew estimates for the laptop were close — the difference between the maximum and the minimum skew estimate was only 0.67 ppm. We also simultaneously measured the clock skew of the laptop and another machine from multiple PlanetLab nodes throughout the world, as well as from a machine of our own with a CDMA-synchronized Dag card [1, 9, 11, 17] for taking network traces with precise timestamps (Table 4). With the exception of the measurements taken by a PlanetLab machine in India (over 300 ms round trip time away), for each experiment, all the fingerprinters (in North America, Europe, and Asia) reported skew estimates within only 0.56 ppm of each other. These experiments suggest that, except for extreme cases, the results of our clock skew estimation techniques are independent of access technology and topology.

Toward understanding the distribution of clock skews across machines, we applied the TCP timestamps technique to the devices in a trace collected on one of the U.S.'s Tier 1 OC-48 links (Figure 2). We also measured the clock skews of 69 (seemingly) identical Windows XP SP1 machines in one of our institution's undergraduate computing facilities (Figure 3). The latter experiment, which ran for 38 days, as well as other experiments, show that the clock skew estimates for any given machine are approximately constant over time, but that different machines have detectably different clock skews. Lastly, we use the results of these and other experiments to argue that the amount of data (packets and duration of data) necessary to perform our skew estimation techniques is low, though we do not perform a rigorous analysis of exactly what "low" means.

**APPLICATIONS AND ADDITIONAL EXPERIMENTS.** To test the applicability of our techniques, we applied our techniques to a honeyd [24] virtual honeynet consisting of 100 virtual Linux 2.4.18 hosts and 100 virtual Windows XP SP1 hosts. Our experiments showed with overwhelming probability that the TCP flows and ICMP timestamp responses were all handled by a single machine as opposed to 200 different machines. We also applied our techniques to a network of five virtual machines running under VMware Workstation [4] on a single machine. In this case, the clock skew estimates of the virtual machines are significantly different from what one would expect from real machines (the skews were large and *not* constant over time; Figure 5). An

application of our techniques, or natural extensions, might therefore be to remotely detect virtual honeynets.

Another applications of our techniques is to count the number of hosts behind a NAT, even if those hosts use random or constant IP IDs to counter Bellovin's attack [7], even if all the hosts run the same operating system, and even if not all of the hosts are up at the same time. Furthermore, when both our techniques and Bellovin's techniques are applicable, we expect our approach to provide a much higher degree of resolution. One could also use our techniques for forensics purposes, e.g., to argue whether or not a given laptop was connected to the Internet from a given access location. One could also use our techniques to help track laptops as they move, perhaps as part of a Carnivore-like project (here we envision our skew estimates as one important component of the tracking; other components could be information gleaned from existing operating system fingerprinting techniques, usage characteristics, and other heuristics). One can also use our techniques to catalyze the unanonymization of prefix-preserving anonymized network traces [28, 29].

**BACKGROUND AND RELATED WORK.** It has long been known that seemingly identical computers can have disparate clock skews. The NTP [19] specification describes a method for reducing the clock skews of devices' system clocks, though over short periods of time an NTP-synchronized machine may still have slight clock skew. In 1998 Paxson [22] initiated a line of research geared toward eliminating clock skew from network measurements, and one of the algorithms we use is based on a descendent of the Paxson paper by Moon, Skelly, and Towsley [20]. Further afield, though still related to clock skews, Pásztor and Veitch [21] have created a software clock on a commodity PC with high accuracy and small clock skew. One fundamental difference between our work and previous work is our goal: whereas all previous works focus on creating methods for eliminating the effects of clock skews, our work exploits and capitalizes on the effects of clock skews.

Anagnostakis et. al. [6] use ICMP Timestamp Requests to study router queuing delays. It is well known that a network card's MAC address is supposed to be unique and therefore could serve as a fingerprint of a device assuming that the adversary can observe the device's MAC address and that the owner of the card has not changed the MAC address. The main advantage of our techniques over a MAC address-based approach is that our techniques are mountable by adversaries thousands of miles and multiple hops away. One could also use cookies or any other persistent data to track a physical device, but such persistent data may not always be available to an adversary, perhaps because the user is privacy-conscious and tries to minimize storage and transmission of such data, or because the user never communicates that data unencrypted.

See [15] for the full version of this paper.

## 2 Clocks and clock skews

When discussing clocks and clock skews, we build on the nomenclature from the NTP specification [19] and from Paxson [22]. A *clock*  $C$  is designed to represent the amount of time that has passed since some initial time  $i[C]$ . Clock  $C$ 's *resolution*,  $r[C]$ , is the smallest unit by which the clock can be incremented, and we refer to each such increment as a *tick*. A resolution of 10 ms means that the clock is designed to have 10 ms granularity, not that the clock is always incremented *exactly* every 10 ms. Clock  $C$ 's *intended frequency*,  $\text{Hz}[C]$ , is the inverse of its resolution; e.g., a clock with 10 ms granularity is designed to run at 100 Hz. For all  $t \geq i[C]$ , let  $R[C](t)$  denote the time reported by clock  $C$  at time  $t$ , where  $t$  denotes the true time as defined by national standards. The *offset* of clock  $C$ ,  $\text{off}[C]$ , is the difference between the time reported by  $C$  and the true time, i.e.,  $\text{off}[C](t) = R[C](t) - t$  for all  $t \geq i[C]$ . A clock's *skew*,  $s[C]$ , is the first derivative of its offset with respect to time, where we assume for simplicity of notation that  $R[C]$  is a differentiable function in  $t$ . We report skew estimates in microseconds per second ( $\mu\text{s}/\text{s}$ ) or, equivalently, parts per million (ppm). As we shall show, and as others have also concluded [22, 20, 26], it is often reasonable to assume that a clock's skew is constant. When the clock in question is clear from context, we shall remove the parameter  $C$  from our notation; e.g.,  $s[C]$  becomes  $s$ .

A given device can have multiple, possibly independent, clocks. For remote physical device fingerprinting, we exploit two different clocks: the clock corresponding to a device's *system time*, and a clock internal to a device's TCP network stack, which we call the device's *TCP timestamps option clock* or *TSopt clock*. We do not consider the hardware bases for these clocks here since our focus is not on understanding why these clocks have skews, but on exploiting the fact these clocks can have measurable skews on popular current-generation systems.

**THE SYSTEM CLOCK.** To most users of a computer system, the most visible clock is the device's *system clock*,  $C_{\text{sys}}$ , which is designed to record the amount of time since 00:00:00 UTC, January 1, 1970. Although the system clocks on professionally administered machines are often approximately synchronized with true time via NTP [19] or, less accurately, via SNTP [18], we stress that it is much less likely for the system clocks on non-professionally managed machines to be externally synchronized. This lack of synchronization is because the default installations of most of the popular operating systems that we tested *do not* synchronize the hosts' system clocks with true time or, if they do, they do so only infrequently. For example, default Windows XP Professional installations only synchronize their system times with Microsoft's NTP server when they boot and once a week thereafter. Default Red Hat 9.0 Linux

installations do not use NTP by default, though they do present the user with the option of entering an NTP server. Default Debian 3.0, FreeBSD 5.2.1, and OpenBSD 3.5 systems, at least under the configurations that we selected (e.g., "typical user"), do not even present the user with the option of installing `ntpd`. For such a non-professionally-administered machine, if an adversary can learn the values of the machine's system clock at multiple points in time, the adversary will be able to infer information about the device's *system clock skew*,  $s[C_{\text{sys}}]$ .

**THE TCP TIMESTAMPS OPTION CLOCK.** RFC 1323 [13] specifies the *TCP timestamps option* to the TCP protocol. A TCP flow will use the TCP timestamps option if the network stacks on both ends of the flow implement the option and if the initiator of the flow includes the option in the initial SYN packet. All modern operating systems that we tested implement the TCP timestamps option. Of the systems we tested, Microsoft Windows 2000 and XP are the only ones that do not include the TCP timestamps option in the initial SYN packet (Microsoft Windows Pocket PC 2002 does include the option when initiating TCP flows). In Section 3 we introduce a trick for making Windows 2000- and XP-initiated flows use the TCP timestamps option.

For physical device fingerprinting, the most important property of the TCP timestamps option is that if a flow uses the option, then a portion of the header of each TCP packet in that flow will contain a 32-bit timestamp generated by the creator of that packet. The RFC does not dictate what values the timestamps should take, but does say that the timestamps should be taken from a "virtual clock" that is "at least approximately proportional to real time [13];" the RFC 1323 PAWS algorithm does stipulate (Section 4.2.2) that the resolution of this virtual clock be between 1 ms and 1 second. We refer to this "virtual clock" as the device's *TCP timestamps option clock*, or its *TSopt clock*  $C_{\text{tcp}}$ . There is no requirement that a device's TSopt clock and its system clock be correlated. Moreover, for popular operating systems like Windows XP, Linux, and FreeBSD, a device's TSopt clock may be unaffected by adjustments to the device's system clock via NTP. To sample some popular operating systems, standard Red Hat 9.0 and Debian 3.0 Linux distributions<sup>2</sup> and FreeBSD 5.2.1 machines have TSopt clocks with 10 ms resolution, OS X Panther and OpenBSD 3.5 machines have TSopt clocks with 500 ms resolution, and Microsoft Windows 2000, XP, and Pocket PC 2002 systems have TSopt clocks with 100 ms resolution. Most systems reset their TSopt clock to zero upon reboot; on these systems  $i[C_{\text{tcp}}]$  is the time at which the system booted. If an adversary can learn the values of a device's TSopt clock at multiple points in time, then the adversary may be able to infer information about the device's *TSopt clock skew*,  $s[C_{\text{tcp}}]$ .

<sup>2</sup>We do not generalize this to all Linux distributions since Knoppix 3.6, with the 2.6.7 experimental kernel, has 1 ms resolution.



### 3 Exploiting the TCP Timestamps Option

In this section we consider (1) how an adversary might obtain samples of a device's TSOpt clock at multiple points in time and (2) how an adversary could use those samples to fingerprint a physical device. We assume for now that there is a one-to-one correspondence between physical devices and IP addresses, and defer to Section 8 a discussion of how to deal with multiple active hosts behind a NAT; in this section we do consider NATs with a single active device behind them.

**THE MEASURER.** The measurer can be any entity capable of observing TCP packets from the fingerprintee, assuming that those packets have the TCP timestamps option enabled. The measurer could therefore be the fingerprintee's ISP, or any tap in the middle of the network over which packets from the device travel; e.g., we apply our techniques to a trace taken on a major Tier 1 ISP's backbone OC-48 links. The measurer could also be any system with which the fingerprintee frequently communicates; prime examples of such systems include a search engine like Google, a news website, and a click-through ads service that displays content on a large number of websites. If the measurer is active, then the measurer could also be the one to initiate a TCP flow with the fingerprintee, assuming that the device is reachable and has an open port. If the measurer is semi-passive or active, then it could make the flows that it observes last abnormally long, thereby giving the measurer samples of the fingerprintee's clock over extended periods of time.

**A TRICK FOR MEASURING WINDOWS 2000 AND XP MACHINES.** We seek the ability to measure TSOpt clock skews of Windows 2000 and XP machines even if those machines are behind NATs and firewalls. But, because of the nature of NATs and firewalls, in these cases we will typically be limited to analyzing flows initiated by the Windows machines. Unfortunately, because Windows 2000 and XP machines do not include the TCP timestamps option in their initial SYN packets, the TCP timestamps RFC [13] mandates that *none* of the subsequent packets in Windows-initiated flows can include the TCP timestamps option. Thus, assuming that all parties correctly implement the TCP RFCs, a passive adversary will not be able to exploit the TCP timestamps option with Windows 2000/XP-initiated flows.

If the adversary is semi-passive, we observe the following trick. Assume for simplicity that the adversary is the device to whom the Windows machine is connecting. After receiving the initial SYN packet from the Windows machine, the adversary will reply with a SYN/ACK, but the adversary will *break the RFC 1323 specification and include the TCP timestamps option in its reply*. After receiving such a reply, our Windows 2000 and XP machines ignored the fact that

they did not include the TCP timestamps option in their initial SYN packets, and included the TCP timestamps option in all of their subsequent packets. As an extension, we note that the adversary does not have to be the device to whom the Windows machine is connecting. Rather, the adversary simply needs to be able to mount a "device-in-the-middle" attack and modify packets such that the Windows machine receives one with the TCP timestamps option turned on. If the adversary is the device's ISP, then the ISP could rewrite the Windows machine's initial SYN packets so that they include the TCP timestamps option. The SYN/ACKs from the legitimate recipients will therefore have the TCP timestamps option enabled and, from that point forward, the Windows machine will include the TCP timestamps option in all subsequent packets in the flows.

We applied this technique to Windows XP machines on a residential cable system with a LinkSys Wireless Access Point and a NAT, as well as to Windows XP SP2 machines using the default XP SP2 firewall, and to Windows XP SP1 machines with the Windows ZoneAlarm firewall. (While current firewalls do not detect this trick, it is quite possible that future firewalls might.)

**ESTIMATING THE TSOPT CLOCK SKEW.** Let us now assume that an adversary has obtained a trace  $\mathcal{T}$  of TCP packets from the fingerprintee, and let us assume for simplicity that all  $|\mathcal{T}|$  packets in the trace have the TCP timestamps option enabled. Toward estimating a device's TSOpt clock skew  $s[C_{\text{tcp}}]$  we adopt the following additional notation. Let  $t_i$  be the time in seconds at which the measurer observed the  $i$ -th packet in  $\mathcal{T}$  and let  $T_i$  be the  $C_{\text{tcp}}$  timestamp contained within the  $i$ -th packet. Define

$$\begin{aligned}x_i &= t_i - t_1 \\v_i &= T_i - T_1 \\w_i &= v_i/\text{Hz} \\y_i &= w_i - x_i \\O_{\mathcal{T}} &= \{(x_i, y_i) : i \in \{1, \dots, |\mathcal{T}|\}\}.\end{aligned}$$

The unit for  $w_i$  is seconds;  $y_i$  is the *observed offset* of the  $i$ -th packet;  $O_{\mathcal{T}}$  is the *offset-set* corresponding to the trace  $\mathcal{T}$ . We discuss below how to compute Hz if it is not known to the measurer in advance. As an example, Figure 1 shows the offset-sets for two devices in a two-hour trace of traffic from an Internet backbone OC-48 link on 2004-04-28 (we omit IP addresses for privacy reasons). Shifting the clocks by  $t_1$  and  $T_1$  for  $x_i$  and  $v_i$  is not necessary for our analysis but makes plots like in Figure 1 cleaner.

If we could assume that the measurer's clock is accurate and that the  $t$  values represent true time, and if we could assume that there is no delay between when the fingerprintee generates the  $i$ -th packet and when the measurer records the  $i$ -th packet, then  $y_i = \text{off}(x_i + t_1)$ . Under these assumptions, and if we make the additional assumption that

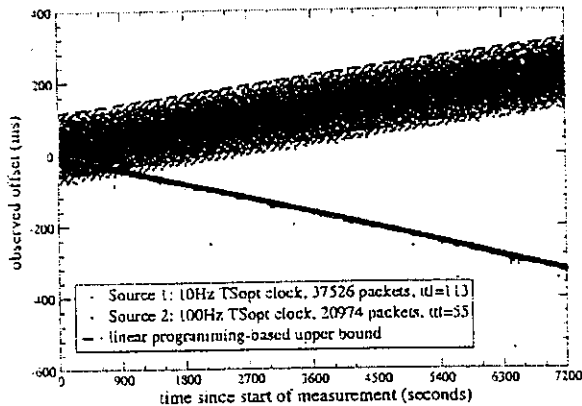


Figure 1. TSopt clock offset-sets for two sources in  $BB_N$ . Trace recorded on an OC-48 link of a U.S. Tier 1 ISP, 2004-04-28 19:30–21:30PDT. The source with the wide band has a 10 Hz TSopt clock, the source with the narrow band has a 100 Hz TSopt clock. A source with no clock skew would have a horizontal band.

$R$  is differentiable, then the first derivative of  $y$ , which is the slope of the points in  $\mathcal{O}_T$ , is the skew  $s$  of  $C_{tcp}$ . Since we cannot generally make these assumptions, we are left to approximate  $s$  from the data.

Let us consider plots like those in Figure 1 more closely. We first observe that the large band corresponds to a device where the TSopt clock has low resolution ( $r = 100$  ms) and that the narrow band corresponds to a device with a higher resolution ( $r = 10$  ms). The width of these bands, and in particular the wide band, means that if the duration of our trace is short, we cannot always approximate the slope of the points in  $\mathcal{O}_T$  by computing the slope between any two points in the set. Moreover, as Paxson and others have noted in similar contexts [22, 20], variable network delay renders simple linear regression insufficient. Consequently, to approximate the skew  $s$  from  $\mathcal{O}_T$ , we borrow a linear programming solution from Moon, Skelly, and Towsley [20], which has as its core Graham’s convex hull algorithm on sorted data [12].

The linear programming solution outputs the equation of a line  $\alpha x + \beta$  that upper-bounds the set of points  $\mathcal{O}_T$ . We use an upper bound because network and host delays are all positive. The slope of the line,  $\alpha$ , is our estimate of the clock skew of  $C_{tcp}$ . In detail, the linear programming constraints for this line are that, for all  $i \in \{1, \dots, |T|\}$ ,

$$\alpha \cdot x_i + \beta \geq y_i,$$

which means that the solution must upper-bound all the points in  $\mathcal{O}_T$ . The linear programming solution then mini-

mizes the average vertical distance of all the points in  $\mathcal{O}_T$  from the line; i.e., the linear programming solution is one that minimizes the objective function

$$\frac{1}{|T|} \cdot \sum_{i=1}^{|T|} (\alpha \cdot x_i + \beta - y_i).$$

Although one can solve the above using standard linear programming techniques, as Moon, Skelly, and Towsley [20] note, there exist techniques to solve linear programming problems in two variables in linear time [10, 16]. We use a linear time algorithm in all our computations.

It remains to discuss how to infer Hz if the measurer does not know it in advance. One solution involves computing the slope of the points

$$\mathcal{I} = \{(x_i, v_i) : i \in \{1, \dots, |T|\}\}$$

and rounding to the nearest integer. One can compute the slope of this set by adapting the above linear programming problem to this set.

AN EQUIVALENT VIEW. If  $A$  is the slope of the points in the above set  $\mathcal{I}$ , derived using the linear programming algorithm, then one could also approximate the skew of  $C_{tcp}$  as  $A/\text{Hz} - 1$ . This approach is simply a different way of arriving at the same solution since we can prove that, when using the linear programming method for slope estimation, both approaches produce the same skew estimate. We use the offset-set approach since these sets naturally yield figures where the skews are clearly visible; e.g., Figure 1.

## 4 Exploiting ICMP Timestamp Requests

THE MEASURER. To exploit a device’s system time clock skew, the measurer could be any website with which the fingerprintee communicates, or any other device on the Internet provided that the measurer is capable of issuing ICMP Timestamp Requests (ICMP message type 13) to the fingerprintee. The measurer must also be capable of recording the fingerprintee’s subsequent ICMP Timestamp Reply messages (ICMP message type 14). In order for this technique to be mountable, the primary limitation is that the device must not be behind a NAT or firewall that filters ICMP.

ESTIMATING THE SYSTEM CLOCK SKEW. Let us now assume that an adversary has obtained a trace  $\mathcal{T}$  of ICMP Timestamp Reply messages from the fingerprintee. The ICMP Timestamp Reply messages will contain two 32-bit values generated by the fingerprintee. The first value is the time at which the corresponding ICMP Timestamp Request packet was received, and the second value is the time at which the ICMP Timestamp Reply was generated; here time is according to the fingerprintee’s system clock,  $C_{sys}$ , and is reported in milliseconds since midnight UTC. Windows machines report the timestamp in little endian for-

mat, whereas all the other machines that we tested report the timestamp in big endian notation. The remaining notation and the method for skew estimation is now identical to what we presented in Section 3, with two minor exceptions. First, the adversary does not have to compute Hz since RFC 792 [23] requires that Hz be 1000 (or, if not, that a special bit be set to indicate non-compliance). Second, since the time reported in the ICMP Timestamp Reply is in milliseconds since midnight UTC, we expect the timestamps reported in the ICMP Timestamp Reply messages to reset approximately once a day; we adjust the  $v$  values accordingly. (The only thing special that our attack exploits about ICMP is the fact that ICMP has a message type that will reveal a device's system time; our techniques would work equally well with any other protocol that leaks information about a device's system or other clock.)

**BRIEF COMPARISON WITH TCP TIMESTAMPS.** For much of the rest of this paper, we focus on our TCP timestamps-based approach for physical device fingerprinting rather than our ICMP-based approach. This is not because we consider the ICMP-based approach to be inferior. Rather, we focus on the TCP timestamps-based approach because most systems have TSopt clocks that operate at a lower frequency than the 1000 Hz clocks included in the ICMP timestamp reply messages. This means that it should require less data for an active adversary to mount our ICMP fingerprinting technique than to mount our TCP timestamps technique. Our positive results for the TCP timestamps-based fingerprinting techniques imply that the ICMP-based fingerprinting technique will be effective and will have low data requirements. Focusing on our TCP timestamps based approach also allows us to experiment with machines behind NATs and firewalls. We also remark that for popular operating systems, if a system does not externally synchronize its system time, then the system's TSopt and system clocks will be highly correlated (Section 7), which means that the distribution of system clock skews for machines not using NTP will be similar to the distribution of TSopt clocks skews.

## 5 Distribution and stability of TSopt clock skew measurements

We now address two fundamental properties that must hold in order for remote clock skew estimation to be an effective physical device fingerprinting technique. First, we show that there is variability in different devices' clock skews, meaning that it is reasonable to expect different devices on the Internet to have measurably different clock skews. Second, we give evidence to suggest that clock skews, as measured by our techniques, are relatively constant over time. These two facts provide the basis for our

min pkts per hour (mp)	min duration per hour (md, mins)	total sources ( $ S $ )	sources with stable skews ( $ S' $ )	entropy (bits)
0	10	18335	8225	4.87
0	30	13517	6859	5.39
0	50	7246	4120	5.87
500	10	4356	2583	5.99
500	30	4016	2446	6.11
500	50	3368	2104	6.18
2000	10	1730	1116	6.22
2000	30	1629	1077	6.32
2000	50	1489	1009	6.41

Table 2. Entropy estimates from BB-2004-04-28 when  $p_v = 1$  ppm.

use of remote clock skew estimation as a physical device fingerprinting technique since they imply that an adversary can gain (sometimes significant) information by applying our techniques to measure a device's or set of devices' clock skews.

The novelty here is not in claiming that these properties are true. Indeed, it is well known that different computer systems can have different clock skews, and others [22, 20, 21, 26] have argued that a given device generally has a constant clock skew. Rather, the contribution here is showing that these properties survive our remote clock skew estimation techniques and, in the case of our analyses of the distribution of clock skews, measuring the bits of information (entropy) a passive adversary might learn by passively measuring the TSopt clock skews of fingerprints.

**DISTRIBUTION OF CLOCK SKEWS: ANALYSIS OF PASSIVE TRACES.** Our first experiment in this section focuses on understanding the distribution of clock skews across devices as reported by our TCP timestamps-based passive fingerprinting technique. For this experiment we analyzed a passive trace of traffic in both directions of a major OC-48 link; CAIDA collected the trace between 19:30 and 21:30 PDT on 2004-04-28. Since the OC-48 link runs North-South, let  $BB_N$  denote the Northbound trace, and let  $BB_S$  denote the Southbound trace (BB stands for backbone). CAIDA obtained the traces using different Dag [11] cards in each direction; these cards' clocks were synchronized with each other, but not with true time. This latter property does not affect the following discussion because (1) the clock skews of the Dag cards appear to be constant and therefore only shift our skew estimates by a constant amount and (2) here we are only interested in the general distribution of the clock skews of the sources in the traces.

Let mp and md be positive integers. For simplicity, fix  $BB = BB_N$  or  $BB_S$ . Also assume for simplicity that BB

only contains TCP packets with the TCP timestamps option turned on. Recall that the trace BB last for two hours. At a high-level, our analysis considers the set  $\mathcal{S}$  of sources in BB that have  $\geq mp$  packets in both the first and the second hours, and where the differences in time between the source's first and last packets in each hour are  $\geq md$  minutes. If  $mp$  and  $md$  are large, then the sources in  $\mathcal{S}$  all generate a large number of packets, and over a long period of time.

For each source in  $\mathcal{S}$ , we apply our clock skew estimation technique from Section 3 to the full trace, the first hour only, and the second hour only. Let  $pv$  be a positive number, and let  $\mathcal{S}'$  be the subset of  $\mathcal{S}$  corresponding to the sources whose skew estimates for the full trace, the first hour, and the second hour are all within  $pv$  ppm of each other, and whose intended frequency Hz is one of the standard values (1, 2, 10, 100, 512, 1000). If  $pv$  is small, then we are inclined to believe that the skew estimates for the sources in  $\mathcal{S}'$  closely approximate the true skews of the respective sources. Table 2 shows values of  $|\mathcal{S}|$  and  $|\mathcal{S}'|$  for different values of  $mp$  and  $md$  and when  $pv = 1$  ppm.

The value  $|\mathcal{S}'|/|\mathcal{S}|$  gives an indication of the ratio of sources of which we can accurately (within  $pv$  ppm) measure the clock skew. While useful, this value provides little information about the actual distribution of the clock skew estimates. Much more (visually) telling are images such as Figure 2, which shows a histogram of the skew estimates (for the full two hour trace) for all the sources in  $\mathcal{S}'$  when  $mp = 2000$ ,  $md = 50$  minutes, and  $pv = 1$  ppm. (The true histogram may be shifted horizontally based on the clock skew of the Dag cards, but a horizontal shift does not affect the general shape of the distribution.) Empirically, for any given values for  $mp$ ,  $md$ , and  $pv$ , we can compute the entropy of the distribution of clock skews. Doing so serves as a means of gauging how many bits of information an adversary might learn by passively monitoring a device's clock skew, assuming that devices' clock skews are constant over time, which is something we address later. To compute the entropy, we consider bins of width  $pv$ , and for each source  $s$  in  $\mathcal{S}'$ , we increment the count of the bin corresponding to devices with clock skews similar to the skew of  $s$  (here we use the skew estimate computed over full two hours). We then allocate another bin of size  $|\mathcal{S}| - |\mathcal{S}'|$ ; this bin counts the number of sources that do not have consistent clock skew measurements. We apply the standard entropy formula [25] to compute the entropy of this distribution of bins, the results of which appear in the last column of Table 2. As one might expect, the amount of information available to an adversary increases as  $mp$  and  $md$  increase.

Assuming that clock skews are constant over time, our results suggest that a passive adversary could learn at least six bits of information about a physical device by applying our techniques from Section 3. More bits of information

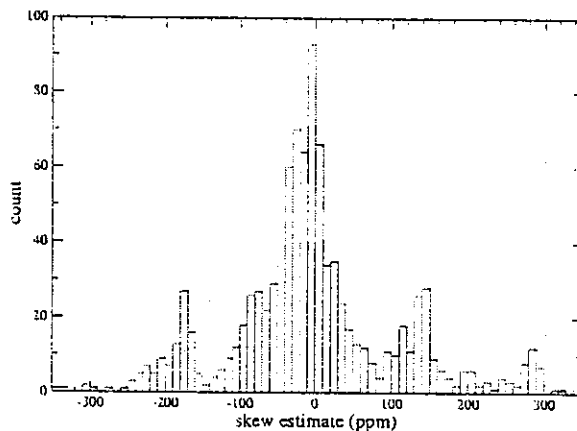


Figure 2. Histogram of TSOpt clock skew estimates for sources in  $BB_N$ . Trace recorded on an OC-48 link of a U.S. Tier 1 ISP, 2004-04-28 19:30–21:30PDT. Here  $mp = 2000$  packets,  $md = 50$  minutes, and  $pv = 1$  ppm.

should be available to an active adversary since an active adversary might be able to force the fingerprintee to send packets more frequently or over longer periods of time.

**DISTRIBUTION OF CLOCK SKEWS: EXPERIMENTS WITH A HOMOGENEOUS LAB.** One observation on the above analysis is that we applied it to a wide variety of machines, which likely ran a wide variety of operating systems. Therefore, one may wonder whether the distribution shown in Figure 2 is due to operating system differences or to actual physical differences on the devices. For example, given only the above results, it might still be possible to argue that if we applied our skew estimates to a large number of (apparently) homogeneous machines, we would get back approximately the same (i.e., indistinguishable) skew estimates for all of the machines. To address this issue, we conducted an experiment with 69 (apparently) homogeneous machines in one of UCSD's undergraduate computing laboratories. All the machines were Micron PCs with 448MHz Pentium II processors running Microsoft Windows XP Professional Service Pack 1. Our measurer, *host2*, was a Dell Precision 410 with a 448MHz Pentium III processor and running Debian 3.0 with a recompiled 2.4.18 kernel; *host2* is located within the University's computer science department and is 3 hops and a half a millisecond away from the machines in the undergraduate laboratory.

To create the requisite trace of TCP packets from these machines, we repeatedly opened and then closed connections from *host2* to each of these machines. Each open-then-close resulted in the Windows machines sending two packets to *host2* with the TCP timestamps option turned on (the Windows machine sent three packets for each flow,

but the TCP timestamp was always zero in the first of these three packets). Because of our agreement with the administrators of these machines, we were only able to open and close connections with these Windows machines at random intervals between zero and five minutes long. Thus, on average we would expect to see each machine send `host2` 48 TCP packets with the TCP timestamps option turned on per hour. The experiment lasted for 38 days, beginning at 19:00PDT 2004-09-07 and ending at approximately 20:30PDT 2004-10-15.

Figure 3 shows a plot, similar to Figure 1, for the 69 Micron machines as measured by `host2` but sub-sampled to one out of every two packets. Note that the plot uses different colors for the observed offsets for different machines (colors are overloaded). Since the slopes of the sets of points for a machine corresponds to the machine's skew, this figure clearly shows that different machines in the lab have measurably different clock skews. Thus, we can easily distinguish some devices by their clock skews (for other devices, we cannot). Because Windows XP machines reset their TSopt clocks to zero when they reboot, some of the diagonal lines seem to disappear several days into the figure. Our algorithms handle reboots by recalibrating the initial observed offset, though this recalibration is not visible in Figure 3. The time in Figure 3 begins on 8:30PDT 2004-09-10 (Friday) specifically because the administrators of the lab tend to reboot machines around 8:00PDT, and beginning the plot on Friday morning means that there are fewer reboots in the figure. We consider this experiment in more detail below, where our focus is on the stability of our clock skew estimates.

**STABILITY OF CLOCK SKEWS.** We now consider the stability of the TSopt clock skews for the devices in the above-mentioned undergraduate laboratory. Consider a single machine in the laboratory. We divide the trace for this machine into 12- and 24-hour periods, discarding 12-hour periods with less than 528 packets from the device, and discarding 24-hour periods with less than 1104 packets from the device (doing so corresponds to discarding 12-hour periods when the device is not up for at least approximately 11 hours, and discarding 24-hour periods that the device is not up for at least 23 hours). We compute the device's clock skew for each non-discarded period, and then compute the difference between the maximum and minimum estimates for the non-discarded periods. This value gives us an indication of the stability of the device's clock skew.

For 12-hour periods, the maximum difference for a single device in the lab ranged between 1.29 ppm and 7.33 ppm, with a mean of 2.28 ppm. For 24-hour periods, the maximum difference for a single device ranged between 0.01 ppm and 5.32 ppm, with a mean of 0.71 ppm. Interestingly, there seems to have been some administrator function at 8:00PDT on 2004-09-10 that slightly adjusted

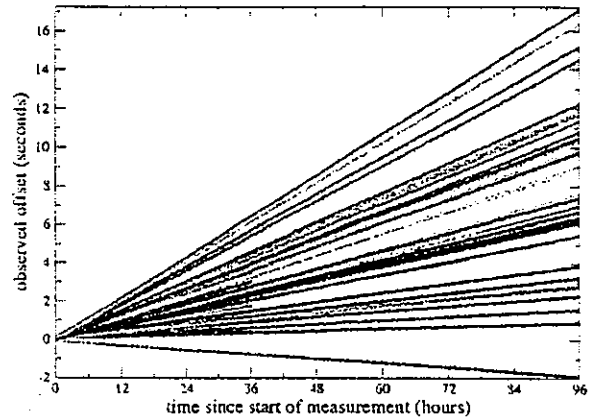


Figure 3. TSopt clock offset-sets for 69 Micron 448MHz Pentium II machines running Windows XP Professional SP1. Trace recorded on `host2`, three hops away, 2004-09-10 08:30PDT to 2004-09-14 08:30PDT.

the TSopt clock skews of some of the machines. If we conduct the same analysis for the trace beginning at 8:30PDT 2004-09-10 and ending on 2004-10-15, for 24-hour periods, the range for maximum difference for each device in the lab dropped to between 0.00 ppm and 4.05 ppm. See [15] for a detailed table.

The current results strongly support our claim that modern processors have relatively stable clock skews. Moreover we believe that if the administrators of the lab allowed us to exchange more packets with the 69 fingerprintees, we would have found the clock skews to be even more stable. In Section 6 we apply our clock skew estimates to a single computer at multiple locations and on multiple dates, and the skew estimates again are close (Table 3); our results below further support our claim of the stability of clock skews over time.

## 6 Access technology-, topology-, and measurer-independent measurements

Here we consider our experiments which suggest that clock skew estimates are relatively independent of the fingerprintee's access technology, the topology between the fingerprintee and the measurer, and the measurer's machine.

**LAPTOPS IN MULTIPLE LOCATIONS.** Our first set of experiments along these lines measures `laptop` connected to the Internet via multiple access technologies and locations (Table 3). For all these experiments, `laptop` is a Dell Latitude C810 notebook with a 1.133GHz Pentium III Mobile processor and running a default installation of Red Hat 9.0 (Linux kernel 2.4.20-8). The measurer in all these experi-

Laptop location	Start time (PDT)	Duration	Packets	Wireless	NAT	Skew est.
San Diego, CA, home cable	2004-07-09, 22:00	3 hours	181	Yes, WEP	Yes	-58.17 ppm
SD Supercomputer Center	2004-07-10, 10:00	3 hours	182	Yes	No	-58.00 ppm
CSE Dept, UCSD	2004-07-12, 12:00	3 hours	180	Yes	No	-58.24 ppm
San Diego, CA, home cable	2004-07-12, 21:00	3 hours	180	Yes	Yes	-58.21 ppm
Clinton, CT, home cable	2004-07-26, 06:00	3 hours	182	No	Yes	-58.19 ppm
San Diego, CA, home cable	2004-09-14, 21:00	30 min	1795	Yes	Yes	-58.22 ppm
SD Supercomputer Center	2004-09-22, 12:00	30 min	1765	Yes	Yes	-58.13 ppm
San Diego dialup, 33.6kbps	2004-10-18, 10:00	30 min	1749	No	No	-57.57 ppm
SD Public Library	2004-10-18, 14:45	30 min	946	Yes	Yes	-57.63 ppm

Table 3. TCP timestamps-based skew estimates of laptop running Red Hat Linux 9.0 when connected to host1 from multiple locations and when not running ntpd. The traces were recorded at host1.

ments, host1, is a Dell Precision 340 with a 2GHz Intel Pentium 4 processor located within the UCSD Computer Science and Engineering department and running Debian 3.0 with a recompiled 2.4.18 Linux kernel; host1 is also configured to synchronize its system time with true time via NTP.

For all experiments, we establish a TCP connection between laptop and host1, and then exchange TCP packets over that connection. On host1, we record a trace of the connection using tcpdump. We then use our techniques from Section 3 to estimate the skew of laptop's TSOpt clock. As the horizontal line in Table 3 indicates, we divide our experiments into two sets. In the first set, our experiments last for three hours and exchange one TCP packet every minute (we do this by performing a sleep(60) on host1). For the second set of experiments, the connections last for 30 minutes, and a packet is exchanged at random intervals between 0 and 2 seconds, as determined by a usleep on host1. With few exceptions, the packets from laptop are all ACKs with no data.

We conduct experiments when the laptop is connected to the Internet via residential cable networks on both coasts (Table 3). For our residential experiments, we use a 802.11b wireless connection with 128-bit WEP encryption, a standard (unencrypted) 802.11b wireless connection, and a standard 10Mbps 10baseT wired connection. We also conducted experiments with our laptop connected to the San Diego Supercomputer Center's 802.11b wireless network, from the UCSD Computer Science and Engineering wireless network, and from the San Diego Public Library's wireless network. As the final column in the table shows, the skew estimates are all within a fraction of a ppm of each other. (If we subsample the first set of experiments to one packet every 3 minutes, then the difference between the skew estimates for any two measurements in the first set is at most 0.45 ppm.)

PLANETLAB AND TOPOLOGY QUESTIONS. Although the

above results strongly suggest that skew estimates are independent of access technology, the above experiments do not stress-test the topology between the fingerprinter and the fingerprintee. Therefore, we conducted the following set of experiments. We selected a set of PlanetLab nodes from around the world that reported, via ntptrace, approximately accurate system times. We chose PlanetLab machines located at the University of California at San Diego, the University of California at Berkeley, the University of Washington, the University of Toronto (Canada), Princeton (New Jersey), the Massachusetts Institute of Technology, the University of Cambridge (UK), ETH (Switzerland), IIT (India), and Equinix (Singapore). These PlanetLab machines, along with host1 and (in one case) CAIDA's test machine with a CDMA-synchronized Dag card, served as our fingerprinters. Our fingerprintees were laptop and host1, where laptop was connected both to the SDSC wireless and to the CAIDA wired networks.

For each of our experiments, and for each of our chosen PlanetLab nodes, we created a flow between the node and the fingerprintee. Over each flow our fingerprintee sent one packet at random intervals between 0 and 2 seconds; here the fingerprintee executed usleep with appropriate parameters. We then recorded the flows on the PlanetLab machines using plabdump. On host1 we recorded the corresponding flow using tcpdump. And on the machine with the Dag card we used Coral [14] (that machine was only reachable when laptop was connected directly to CAIDA's wired network). We then computed the skew using the techniques from Section 3. The results for laptop are in Table 4. Notice that the skew estimates are in general within a fraction of a ppm of each other, suggesting that our skew estimates are independent of topology.

For distance measurements for Table 4, we used traceroute to determine hop count, and then used mean time between when tcpdump recorded a packet on the measured device and the time between when plabdump recorded the packet on the measurer. This distance estimate also includes

Measurer	laptop, 2004-09-17, 08:00–10:00 PDT		laptop, 2004-10-08, 08:00–10:00 PDT	
	Skew estimate	Dist. from measurer	Skew estimate	Dist. from measurer
host1	−58.23 ppm	7 hops, 2.77 ms	−58.03 ppm	8 hops, 1.16 ms
San Diego, CA	−58.07 ppm	7 hops, 1.21 ms	−58.03 ppm	8 hops, 1.15 ms
Berkeley, CA	−58.17 ppm	10 hops, 4.02 ms	−58.02 ppm	12 hops, 5.06 ms
Seattle, WA	−58.15 ppm	8 hops, 14.74 ms	−58.01 ppm	9 hops, 15.12 ms
Toronto, Canada	−58.31 ppm	16 hops, 44.43 ms		
Princeton, NJ	−57.97 ppm	13 hops, 37.59 ms	−57.91 ppm	14 hops, 36.97 ms
Boston, MA	−57.93 ppm	12 hops, 35.82 ms	−58.10 ppm	13 hops, 41.09 ms
Cambridge, UK	−58.06 ppm	20 hops, 84.19 ms	−58.18 ppm	21 hops, 86.45 ms
ETH, Switzerland	−58.38 ppm	20 hops, 90.51 ms	−58.40 ppm	21 hops, 84.07 ms
IIT, India			−59.60 ppm	16 hops, 199.27 ms
Equinix, Singapore	−58.10 ppm	18 hops, 99.50 ms	−58.05 ppm	15 hops, 93.55 ms
CAIDA test lab			−57.98 ppm	5 hops, 0.24 ms

Table 4. Skew estimates of laptop, running Red Hat 9.0 with `ntpd`, for traces taken simultaneously at multiple locations. On 2004-09-17 the laptop was connected to the SDSC wireless network, and on 2004-10-08 the laptop was connected to the CAIDA wired network. The Toronto and India lines have empty cells because the PlanetLab machines at those locations were down during the experiment. The Boston machine on 2004-10-08 was a different PlanetLab machine than the one on 2004-09-17. The empty cell for the CAIDA test lab is because the lab is only reachable from CAIDA's wired network.

the time spent in the application layers on the machines, but should give a rough estimate of the time it takes packets to go from the fingerprintee to the measurer.

The results of these experiments suggest that our `TSopt` clock skew estimation technique is generally independent of the topology and distance between the fingerprinter and the fingerprintee. Furthermore, these results suggest that our skew estimation technique is independent of the actual fingerprinter, assuming that the fingerprinter synchronizes its system time with NTP [19] or something better [26].

## 7 Effects of operating system, NTP, and special cases

OPERATING SYSTEMS AND NTP ON FINGERPRINTEE. In Table 5 we show skew estimates for the same physical device, laptop, running both Red Hat 9.0 and Windows XP SP2, and both with and without NTP-based system clock synchronization. (For this experiment, laptop sent one packet to the measurer, host1, at random intervals between 0 and 2 seconds; laptop was connected to the SDSC wireless network, and was 7 hops away from host1; host1 also sent a ICMP Timestamp Request to laptop at random intervals between 0 and 60 seconds.) The table shows that, for the listed operating systems, the system clock and the `TSopt` clock effectively have the same clock skew when the device's system time is not synchronized with NTP, and that the `TSopt` clock skew is independent of whether the device's system clock is maintained via NTP.

Although not shown in the figure, our experiments with OpenBSD 3.5 on another machine suggest that the `TSopt` clock and system clock on default OpenBSD 3.5 installations have the same skew (approximately 68 ppm). On the other hand, at least with this test machine, the `TSopt` clock and system clock on a default FreeBSD 5.2.1 system have different skews (the `TSopt` clock skew estimate is about the same as with OpenBSD, but the system clock skew estimate is approximately 80 ppm). When we turn on `ntpd` under FreeBSD 5.2.1, the `TSopt` clock skew remained unchanged.

POWER OPTIONS FOR LAPTOPS. We also consider how the clock skews of devices are affected by the power options of laptops. In the case of Red Hat 9.0, when laptop is running with the power connected, if we switch to battery power, there is a brief jump in the `TSopt` clock offset-set for the device, and then the device continues to have the same (within a fraction of a ppm) clock skew. For laptop running Windows XP SP2, if the laptop is idle from user input but continues to maintain a TCP flow that we can monitor, then the `TSopt` clock skew changes after we switch to battery power. If we repeat this experiment several times, and if we boot with only battery power, we find that the clock skews with battery power are in all cases similar. When booting with outlet power, the clock skew on laptop running Windows XP initially begins with a large magnitude, and then stabilizes to a skew like that in Table 5 until we disconnect the power; the initially large skew may be due to the laptop recharging its batteries. We have not sampled a large enough set of laptops to determine whether the

Start time	Operating System	NTP	skew estimate (TCP tstamp)	skew estimate (ICMP tstamp)
2004-09-22, 12:00 PDT	Red Hat 9.0	No	-58.20 ppm	-58.16 ppm
2004-09-17, 08:00 PDT	Red Hat 9.0	Yes	-58.16 ppm	-0.14 ppm
2004-09-22, 21:00 PDT	Windows XP SP2	No	-85.20 ppm	-85.42 ppm
2004-09-23, 21:00 PDT	Windows XP SP2	Yes	-84.54 ppm	1.69 ppm

Table 5. Experiments for the same physical device, laptop, running different operating systems and with NTP synchronization both on and off. For all experiments, laptop was located on the SDSC wireless network. Additionally, laptop was up for an hour before the Windows measurements.

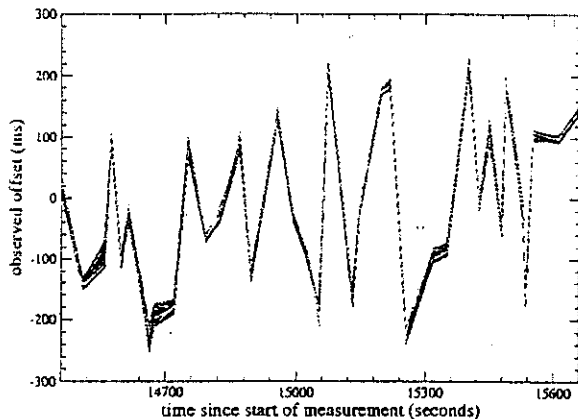


Figure 4. TSopt clock offset-sets for 100 honeyd 0.8b Windows XP SP1 virtual hosts. Start time: 2004-09-19, 23:00PDT; honeyd running on host3. Points are connected in this figure to highlight the correlation between the virtual hosts.

clock skews with battery power are a simple function of the clock skews with outlet power, though the skews with battery power seem to be consistent for a single laptop.

## 8 Applications

We now consider some applications of our techniques, though we emphasize that our most important results are the foundations we introduced in the previous sections that make the following applications possible.

**VIRTUALIZATION AND VIRTUAL HONEYNETS.** We created a honeyd [24] version 0.8b virtual honeynet consisting of 100 Linux 2.4:18 virtual hosts and 100 Windows XP SP1 virtual hosts. Our server in this experiment, host3, is identical to host1, has 1GB of RAM, and maintains its system time via NTP. We ran honeyd with standard nmap and xprobe2 configuration files as input; honeyd used

the information in these files to mimic real Linux and Windows machines. We ran nmap and xprobe2 against the virtual hosts to verify that nmap and xprobe2 could not distinguish the virtual hosts from real machines.

We applied our TCP timestamps- and ICMP-based skew estimation techniques to all 200 virtual hosts. Our fingerprinter in this experiment was on the same local network. We observed several methods for easily distinguishing between honeyd virtual hosts and real machines. First, we noticed that unlike real Linux and Windows machines, the virtual hosts always returned ICMP Timestamp Replies with zero in the transmit timestamp field. Additionally, we observed that the honeyd Windows XP virtual hosts had TSopt clocks  $C_{tcp}$  with  $Hz\{C_{tcp}\} = 2$ , whereas all of the real Windows XP machines that we tested had  $Hz\{C_{tcp}\} = 10$ . The lesson here is that although the nmap and xprobe2 configuration files provide enough information for the respective programs to effectively fingerprint real operating systems, the configuration files do not provide enough information for honeyd to be able to correctly mimic all aspects of the Linux and Windows protocol stacks.

Even if honeyd completely mimicked the network stacks of real Linux 2.4.18 and Windows XP SP1 machines, we could still use our remote physical device fingerprinting techniques to distinguish between our 200 virtual hosts and 200 real machines. Our TSopt clock skew estimates for all 200 virtual hosts were approximately zero and the system clock skew estimates for all 200 virtual hosts were approximately the same positive value. Given our discussion in Section 5 of the distribution of clock skews, this lack of variability in clock skews between virtual hosts is not what one would expect from real machines. Furthermore, the TSopt and system clocks between all the virtual hosts of the same operating system were highly correlated; e.g., Figure 4 shows the TSopt offset-sets for all 100 Windows XP SP1 virtual hosts 241 minutes into our experiment. We communicated our results to the author of honeyd and, in response, version 1.0 of honeyd randomly assigns TSopt clock skews to each virtual host using a Gaussian distribution around the server's system time. This decision may affect other components of the system, e.g., if the server



runs `ntpd`, changes to the server's system time may appear as global changes to the distribution of the virtual hosts' clocks. Version 1.0 of `honeyd` still issues ICMP Timestamp Replies with zero transmit timestamps. Furthermore, the system clocks on version 1.0 `honeyd` virtual hosts are still highly synchronized and are too fast by several orders of magnitude.

To experiment with real virtualization technologies, we installed VMware Workstation 4.5.2 on `host3`, but this time `host3` ran Red Hat 9.0. We then installed five default copies of Red Hat 9.0 under VMware. We applied our skew estimation techniques to these five virtual machines, as well as to `host3`. The results show that the five virtual machines do not have constant (or near constant) clock skews, shown by the non-linearity of the points in Figure 5. Furthermore, the magnitude of the clock skews on these virtual machines is larger than we would expect for physical machines. We feel confident that these observations and natural extensions could prove useful in distinguishing virtual honeynets from real networks.

**COUNTING THE NUMBER OF DEVICES BEHIND A NAT.** Another natural application of our techniques is to count the number of devices behind a NAT. To briefly recall previous work in this area, Bellovin [7] showed that an adversary can exploit the IP ID field to count the number of devices behind a NAT, but his approach is limited in three ways: (1) the IP ID field is only 16-bits long; (2) recent operating systems now use constant or random IP ID fields; and (3) his technique cannot count the total number of devices behind a NAT if not all of them are active at the same time. Our suggested approach to this problem has two phases. First, partition the trace into (candidate) sets corresponding to different sequences of time-dependent TCP timestamps; creating such a partition is relatively easy to do unless two machines have approximately the same TSopt clock values at some point in time, perhaps because the machines booted at approximately the same time. Then apply our clock skew estimation techniques to each partition, counting hosts as unique if they have measurably different clock skews. If two devices have approximately the same TSopt clock values at some point in time but have measurably different clock skews, then one can detect and correct this situation in the analysis of the partition's offset-set.

**FORENSICS AND TRACKING INDIVIDUAL DEVICES.** The utility of our techniques for forensics purposes follows closely from our claims (1) that there is variability in the clock skews between different physical devices (Section 5), (2) that the clock skew for a single device is approximately constant over time (Section 5), and (3) that our clock skew estimates are independent of access technology, topology, and the measurer (Section 6). For forensics, we anticipate that our techniques will be most useful when arguing that a

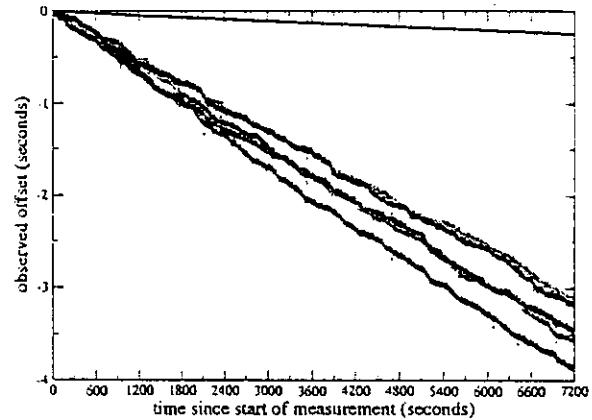


Figure 5. TSopt clock offset-sets for five VMware Workstation virtual machines running Red Hat 9.0, and for the host, `host3`, also running Red Hat 9.0. 2004-10-27 17:00–19:00PDT. The top set of points corresponds to the TSopt clock offset set for `host3`.

given device was not involved in a recorded event. With respect to tracking individual devices, we stress that our techniques do not provide unique serial numbers for devices, but that our skew estimates do provide valuable bits of information that, when combined with other sources of information such as operating system fingerprinting results, can help track individual devices on the Internet.

**UNANONYMIZING ANONYMIZED DATA SETS.** It is common for organizations that provide network traces containing payload data to anonymize the IP addresses in the traces using some prefix-preserving anonymization method [28, 29]. If an organization makes available both anonymized and unanonymized traces from the same link, one can use our techniques to catalyze the unanonymization of the anonymized traces. Such a situation is not hypothetical: in addition to the 2004-04-28 trace that we used in Section 5, CAIDA took another trace from the same link on 2004-04-21, but the 2004-04-21 trace included payload data and was therefore anonymized.

To study how one might use our clock skew estimation techniques to help unanonymize anonymized traces, on 2005-01-13 and 2005-01-21 CAIDA took two two-hour traces from a major OC-48 link (the same link from which CAIDA captured the 2004-04-28 trace). We anonymized the 2005-01-13 trace and experimented with our ability to subsequently unanonymize it. Given the value of a device's TSopt clock and knowledge of that clock's intended frequency Hz, we can compute the approximate uptime of the device. (Prior to our work, one method for inferring Hz from a passive trace would be to use a program like

poE [3].) As a first attempt at unanonymizing the 2005-01-13 trace, we paired anonymized IP addresses from 2005-01-13 with IP addresses from 2005-01-21 when our uptime estimate of a host in 2005-01-21 is eight days higher (plus or minus five minutes) than the uptime of a host in 2005-01-13 and when both hosts have the same TTLs and intended frequencies. Our program produced 4613 pairs of candidate anonymous to real mappings, of which 2660 (57.66%) were correct. To reduce the number of false matches, especially for small uptimes, we modified our program to filter out pairs that have TSOpt clock skews different by more than 3 ppm. We also incorporated our clock skew estimates into our uptime estimates. These changes reduced the number of candidate mappings to 2170, of which 1902 (87.65%) were correct. There are a total of 11862 IP addresses in both the 2005-01-13 and 2005-01-21 traces that have the TCP timestamps option enabled. Since the anonymization is prefix-preserving, given the candidate mappings one can begin to unanonymize address blocks. We are unaware of any previous discussion of the problems to prefix-preserving anonymization caused by leaking information about a source via the TCP timestamps option.

## 9 Other measurement techniques

Although the techniques we describe above will likely remain applicable to current generation systems, we suspect that future generation security systems might try to resist some of the physical device fingerprinting techniques that we uncover. In anticipation of these future systems, we consider possible avenues for clock-based physical device fingerprinting when information about a system's TSOpt clock or system clock is not readily available to an adversary; we do not consider here but recognize the possibility of fingerprinting techniques that profile other aspects of a device's hardware, e.g., processor speed or memory. These directions assume that new operating systems mask or do not include the TSOpt clock values in the TCP headers and do not reply to ICMP Timestamp Requests, but that the systems' underlying clocks still have non-negligible skews. (This assumption may not be valid if, for example, at boot a new operating system does a more precise estimation of the oscillator frequencies supplying the hardware basis for the clocks.) The techniques we propose in this section are less refined than the techniques elsewhere in this paper; we envision them as starting points for more sophisticated techniques.

**FOURIER TRANSFORM.** Some systems send packet at 10 or 100 ms intervals, perhaps due to interrupt processing or other internal operating system feature on one side of a flow. When this condition holds, we can use the Fourier transform to extract information about the system's clock skew. Figure 6 plots the TSOpt clock offset-sets for a device in BB<sub>5</sub> with a 2 Hz TSOpt clock. The five diagonal

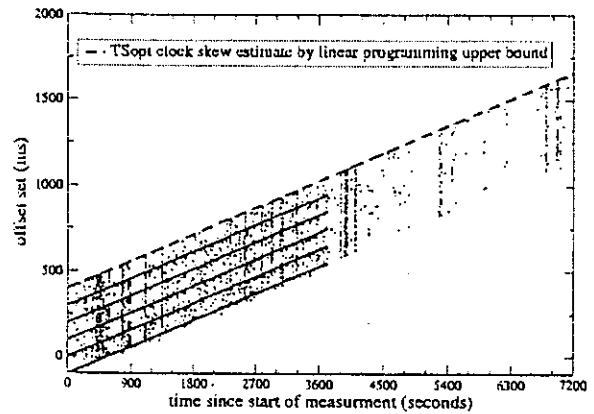


Figure 6. TSOpt clock skew estimate for a source in BB<sub>5</sub>. Trace recorded on an OC-48 link of a U.S. Tier 1 ISP, 2004-04-28 19:30–21:30PDT. TSOpt clock skew estimate via linear programming: 175.2 ppm. Clock skew estimate via the Fourier transform: 175.6 ppm.

bands suggests that the machine clusters packet transmissions at approximately 100 ms intervals, and we can use the Fourier transform on packet arrival times to estimate the frequency at which the device actually transmits packets (here packet arrival times refers to the times at which the monitor records the packets). For the source shown in Figure 6, after computing the Fourier transform, the frequency with the highest amplitude was 25.00439, which implies a skew of  $25.00439/25 - 1$ , or 175.6 ppm. Moreover the top 19 frequencies output by the Fourier transform all imply skews between 171.0 ppm and 179.3 ppm. These values are all close to the 175.2 ppm output by our TCP timestamps-based approach but do not make any use the TCP timestamps contained with the packets.

Although our Fourier-based technique does not require knowledge of a device's TSOpt or system clocks, our Fourier-based solution is currently not automated. This lack of automation, coupled with the fact that current generation systems readily relinquish information about their TSOpt and system clocks, means that our Fourier-based solution is currently less attractive than the techniques we described in Sections 3 and 4.

**PERIODIC USER-LEVEL ACTIVITIES.** Toward estimating the system clock skew of devices that do not synchronize their system times with NTP, we note that many applications perform certain operations at semi-regular intervals. For example, one can configure most mail clients to poll for new mail every  $n$  minutes. As another example, Broido, Nemeth, and claffy show that some Microsoft Windows 2000 and XP systems access DNS servers at regular inter-

vals [8]. It may be possible to infer information about a device's system clock skew by comparing differences between actual intervals of time between these periodic activities and what the application intends for those intervals of time to be.

## 10 Conclusions

In this study we verified the ability and developed techniques for remote physical device fingerprinting that exploit the fact that modern computer chips have small yet non-trivial and remotely detectable clock skews. We showed how our techniques apply to a number of different practically useful goals, ranging from remotely distinguishing between virtual honeynets and real networks to counting the number of hosts behind a NAT. Although the techniques we described will likely remain applicable to current generation systems, we suspect that future generation security systems might offer countermeasures to resist some of the fingerprinting techniques that we uncover. In anticipation of such developments, we discussed possible avenues for physical device fingerprinting when information about a system's TSOPT clock or system clock are not readily available to the adversary. Our results compellingly illustrate a fundamental reason why securing real-world systems is so genuinely difficult: it is possible to extract security-relevant signals from data canonically considered to be noise. This aspect renders perfect security elusive, and even more ominously suggests that there remain fundamental properties of networks that we have yet to integrate into our security models.

## Acknowledgments

We thank Bruce Potter and Stefan Savage for helpful discussions, Emile Aben, Dan Andersen, Colleen Shannon, and Brendan White for collecting some of the traces that we analyzed, and William Griswold for loaning us a PDA from the HP Mobile Technology Solutions gift to the ActiveCampus project. All three authors were supported by the SciDAC program of the US DOE (award # DE-FC02-01ER25466). T. Kohno was also supported by NDSEG and IBM Ph.D. Fellowships.

## References

- [1] Endace measurement systems, 2004. URL: <http://www.endace.com/>.
- [2] Nmap free security scanner, 2004. URL: <http://www.insecure.org/nmap/>.
- [3] Project details for p0f, 2004. URL: <http://freshmeat.net/projects/p0f/>.
- [4] VMware virtual infrastructure, 2004. URL: <http://www.vmware.com/>.
- [5] Xprobe official home, 2004. URL: <http://www.sys-security.com/html/projects/X.html>.
- [6] K. G. Anagnostakis, M. Geenwald, and R. S. Ryger. cing: Measuring network-internal delays using only existing infrastructure. In *INFOCOM*, 2003.
- [7] S. M. Bellovin. A technique for counting NATted hosts. In *IMW*, 2002.
- [8] A. Broido, E. Nemeth, and k. claffy. Spectroscopy of DNS update traffic. In *SIGMETRICS*, 2003.
- [9] S. Donnelly. *High precision timing in passive measurements of data networks*. Ph.D. thesis, University of Waikato, Hamilton, New Zealand, 2002.
- [10] M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13, 1984.
- [11] I. D. Graham, M. Pearson, J. Martens, and S. Donnelly. Dag - A cell capture board for ATM measurement systems. URL: <http://dag.cs.waikato.ac.nz/>.
- [12] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1, 1972.
- [13] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, May 1992.
- [14] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and k. claffy. The architecture of the CoralReef Internet traffic monitoring software suite. In *PAM*, 2001.
- [15] T. Kohno, A. Broido, and k. claffy. Remote physical device fingerprinting. Full version of this paper, available at <http://www-cse.ucsd.edu/users/tkohno/papers/PDF/>, 2005.
- [16] N. Megiddo. Linear-time algorithms for linear programming in  $r^3$  and related problems. *SIAM J. Comput.*, 12, 1983.
- [17] J. Micheel, S. Donnelly, and I. Graham. Precision timestamping of network packets. In *IMW*, 2001.
- [18] D. Mills. Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI. RFC 2030, 1996.
- [19] D. L. Mills. Network time protocol (version 3): Specification, implementation and analysis. RFC 1305, 1992.
- [20] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *INFOCOM*, 1999.
- [21] A. Pásztor and D. Veitch. PC based precision timing without GPS. In *SIGMETRICS*, 2002.
- [22] V. Paxson. On calibrating measurements of packet transit times. In *SIGMETRICS*, 1998.
- [23] J. Postel. Internet control message protocol. RFC 792, 1981.
- [24] N. Provos. A virtual honeypot framework. In *Usenix Security 2004*, 2004.
- [25] C. Shannon. *The mathematical theory of communication*. 1949. Urbana, University of Illinois Press.
- [26] D. Veitch, S. Babu, and A. Pásztor. Robust synchronization of software clocks across the Internet. In *IMC*, 2004.
- [27] F. Veysset, O. Courtay, and O. Heen. New tool and technique for remote operating system fingerprinting, 2002. URL: <http://www.intranode.com/fr/doc/ring-short-paper.pdf>.
- [28] J. Xu, J. Fan, M. Ammar, and S. B. Moon. On the design and performance of prefix-preserving IP traffic trace anonymization. In *IMW*, 2001.
- [29] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *ICNP*, 2002.

# 國立中山大學九十四學年度博士班招生考試試題

科目：資訊科技論文評述(二)【資管系選考】

共 / 頁 第 / 頁

Read the attached paper carefully and answer the following questions.

1. Please write an abstract for this paper in 300 Chinese words. (10%)
2. This paper considers three metrics for site's QoS. For each metric, describe how its measures affect the aggregate metric shown in Equation (1). (10%)
3. For each of the three QoS metric, show how the increase of the maximum number of TCP connections may affect its measures. (10%)
4. When searching for the best configuration parameter setting, this paper reports on using hill-climbing algorithm. How does the hill-climbing algorithm work? (5%)
5. Please criticize this paper. (15%)



# Automatic QoS Control

Daniel A. Menascé • George Mason University • [menasce@cs.gmu.edu](mailto:menasce@cs.gmu.edu)

User sessions, usually consisting of sequences of consecutive requests from customers, comprise most of an e-commerce site's workload. These requests execute e-business functions such as browse, search, register, login, add to shopping cart, and pay.

My colleagues and I proposed and applied a characterization methodology for e-commerce sites that views an e-commerce site's workload in a hierarchical way.<sup>1</sup> The top level is the *session* level, in which statistics of interest include session duration, type, and arrival rate. The next level is the *function* level, in which we're interested in each function type's frequency of occurrence and arrival process, as well as the popularity of certain terms in search requests. Finally, the bottom layer characterizes the workload in terms of HTTP *requests*. Important statistics at this level include the arrival process of HTTP requests and the analysis of long-range dependencies in the arrival stream.

Once we properly understand and characterize a workload, we must assess its effect on the site's quality of service (QoS), which is defined in terms of response time, throughput, the probability that requests will be rejected, and availability (see Figure 1). We can assess an e-commerce site's QoS in many different ways. One approach is by measuring the site's performance, which we can determine from a production site using a real workload or from a test site using a synthetic workload (as in load testing).<sup>2</sup> Another approach consists of using *performance models*.

### Performance Models

A computer system performance model is a way to estimate the value of performance metrics based on a workload model — a synthetic representation of the actual workload — and on the values of the system's configuration parameters (see Figure 2). Examples of configuration parameters include the

maximum number of TCP connections at each Web server and the number of threads at the Web and application servers.

A performance model fulfills several purposes. During an e-commerce site's design phase, for example, it can help compare competing alternatives (for example, should you build a site with a few powerful servers or many low-capacity ones).

Performance models also can help during production for capacity-planning purposes.<sup>3</sup> For medium- to long-term time horizons (on the order of months), such models can help designers predict if any parts of the site won't meet QoS requirements under different scenarios. An example would be how the site's response time might vary when the new marketing campaign (which is expected to double the site's visitors) launches.

Another use of performance models is in the short-term (on the order of minutes) automatic reconfiguration of an e-commerce site to preserve its QoS goals. Many reasons demonstrate the necessity of a site's being able to automatically change its configuration parameters to meet those goals:

- The site's architecture is complex, with multiple tiers of servers, each of which can have many components.<sup>4</sup>
- As Figure 3 shows, an e-commerce site's workload is bursty in nature, especially at fine timescales (for example, minutes).<sup>1</sup>
- Humans cannot react quickly enough to the workload's fast variations to determine the values of the numerous configuration parameters that could optimize the QoS for a given workload's intensity.

Let's look at the approach my colleagues at George Mason and I took that uses performance models in the design and implementation of an automatic QoS controller for e-commerce sites.<sup>5</sup>

## A Dynamic QoS Control Approach

We designed a *QoS controller* that continuously monitors an e-commerce site's workload and determines the configuration that best meets the site's QoS goals (see Figure 4, next page). At regular intervals—called *controller intervals*—the QoS controller executes an algorithm that takes into account the observed workload, the desired QoS levels, and performance data from the site's various resources (such as utilizations of CPUs and disks) to determine the best values of the configuration parameters.

### Metrics

We must first define a metric for the QoS controller to use as an optimization goal. This metric should

- aggregate several individual QoS metrics such as response time, throughput, and probability of rejection in a way that is independent of the units used to measure these metrics;
- be a single number with no units;
- allow the e-commerce site managers to assign different importance levels to the various individual QoS metrics;
- take into account the desired upper or lower bounds of the individual QoS metrics; and
- increase as the value of an individual metric improves with respect to its bound and decrease as the value of an individual metric deteriorates with respect to its bound.

The following metric satisfies these properties:

$$QoS = \sum_{k=1}^n w_k \times f_k(\Delta_k) \quad (1)$$

where  $n$  is the number of QoS metrics being aggregated,  $w_k$  is a relative importance weight assigned by site management to QoS metric  $k$  (all weights sum to 1),  $\Delta_k$  is a relative deviation of the QoS metric  $k$  defined in a way that the relative deviation is positive when the QoS metrics exceeds their goal and negative otherwise, and  $f_k()$  is an increasing function of  $\Delta_k$ .

Expressions for relative deviations of response time ( $R$ ), throughput ( $X$ ), and probability of rejection ( $P$ ), respectively, are

$$\Delta QoS_R = \frac{R_{max} - R_{measured}}{R_{max}} \quad (2)$$

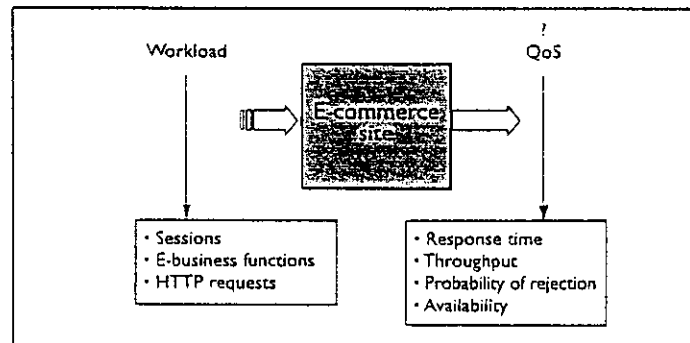


Figure 1. An e-commerce site, its workload, and the resulting QoS metrics. The workload is characterized in terms of sessions, e-business functions, and HTTP requests. The QoS metrics include response time, throughput, probability of request rejection, and availability.

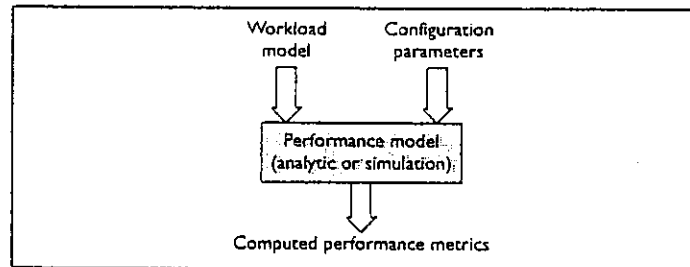


Figure 2. A computer system's performance model. Given a workload model and the values of configuration parameters, a performance model computes the values of the system's performance metrics.

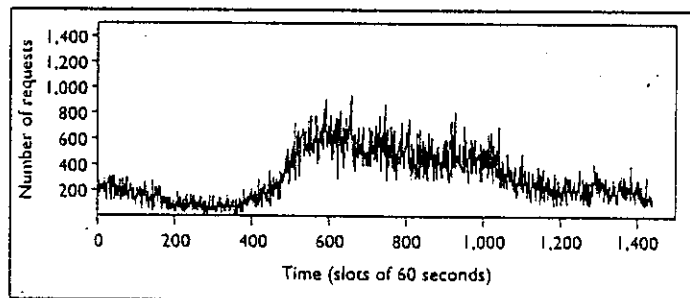


Figure 3. Arrival process of HTTP requests to an online bookstore in one day.<sup>1</sup> The x-axis is divided into time slots of one minute, and the y-axis represents the number of arrivals in each time slot.

$$\Delta QoS_X = \frac{X_{measured} - X_{min}}{X_{min}} \quad (3)$$

$$\Delta QoS_P = \frac{P_{max} - P_{measured}}{P_{max}} \quad (4)$$

### QoS Controller Design

(2) The QoS controller algorithm executes a combina-

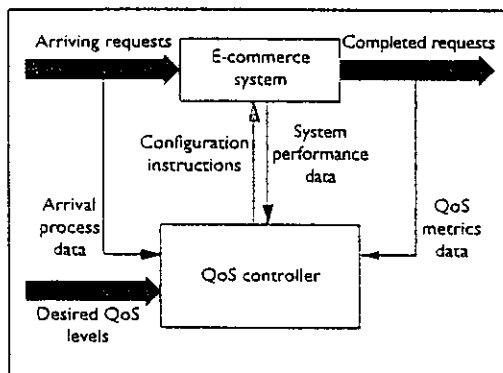


Figure 4. An e-commerce site with a QoS controller. The controller monitors the arrival of requests and the resulting QoS. It also collects system performance data, which the controller uses to determine how the site configuration should be changed.

torial optimization algorithm<sup>6</sup> searching for an optimal point in the space of parameter configuration values. Each point in the search space is represented by a tuple  $(c_1, \dots, c_m)$  where  $c_i (i = 1, \dots, m)$  is the value of the site's  $i$ -th configuration parameter. Combinatorial optimization algorithms traverse the search space in a nonexhaustive manner seeking the point with the optimum or close-to-optimum value of some function associated with each point. In our case, we associate the aggregated QoS metric described in Equation 1 as the function of a point that the algorithm must optimize. The controller algorithm essentially tries to find the configuration tuple that maximizes the QoS metric's value for the workload intensity value observed in the preceding measurement interval or intervals.

Equation 1's QoS metric uses the QoS deviations in Equations 2 through 4, which rely on the lower and upper bounds of the QoS metrics and on measured values for those metrics. During the combinatorial optimization algorithm's execution, we must replace QoS metric measurements with QoS metric predictions for each point of the search space that the algorithm evaluates. This prediction is done through the use of analytic performance models based on *queuing networks* (QNs).<sup>3,4</sup> These models can be solved very efficiently and therefore lend themselves to being used by a combinatorial optimization algorithm to quickly evaluate the quality of each point in the search space.

The QoS controller's QN model must be able to capture the impact of the configuration parameters' values on the site's QoS. Because these parameters include software elements such as number and maximum queue size for threads, the QN

model must be able to capture contention for both software and hardware resources.

Figure 5 shows an example of a QN that models an e-commerce site's software aspects and hardware resources. The top part of the figure shows a Web server process and its various threads. Requests that require the execution of an e-business function (for example, search or add to cart) are sent to a queue of application server threads. If an application server thread needs data from a database, a DB server thread queues that request for service. The figure also illustrates that the various software queues might have limited sizes. When these limits are reached, requests are rejected as indicated by the red arrows in the figure.

The bottom part of Figure 5 shows the various machines used to support the software servers. Each machine has one or more CPUs and one or more disks. This type of QN model takes into account the interaction of software and hardware resources and requires specific solution techniques.<sup>7</sup> Each time the QoS controller algorithm runs, it computes the QN model's parameters by using system performance measurements accumulated during the previous controller interval. Once the QoS controller reaches a desired configuration vector, it must send commands out to the site's various servers to instruct them to change their configurations.

## Conclusion

We experimented with this approach by implementing an e-commerce site compatible with TPC-W<sup>8</sup> and a QoS controller that uses an aggregate metric that included response time, probability of rejection, and throughput. We used the hill-climbing combinatorial optimization algorithm, which is a very simple and easy-to-implement algorithm that at each point selects the neighbor with the best QoS to continue the traversal of the search space. The search stops if no improvement in QoS is obtained or if a maximum number of steps is reached.

Despite the fact that the hill-climbing algorithm could generate local optima instead of moving to a global optimum, it produced very good results. As the site's workload intensity steadily increased, the QoS controller was able to keep the aggregate QoS at much higher levels than when the controller was disabled. Without the controller, the QoS metric became negative, showing that at least one of the individual QoS metrics did not meet its goal. □

## References

1. D.A. Menascé et al., "A Hierarchical and Multiscale Approach to Analyze E-Business Workloads," to be published in *Performance Evaluation*, North-Holland, 2002.

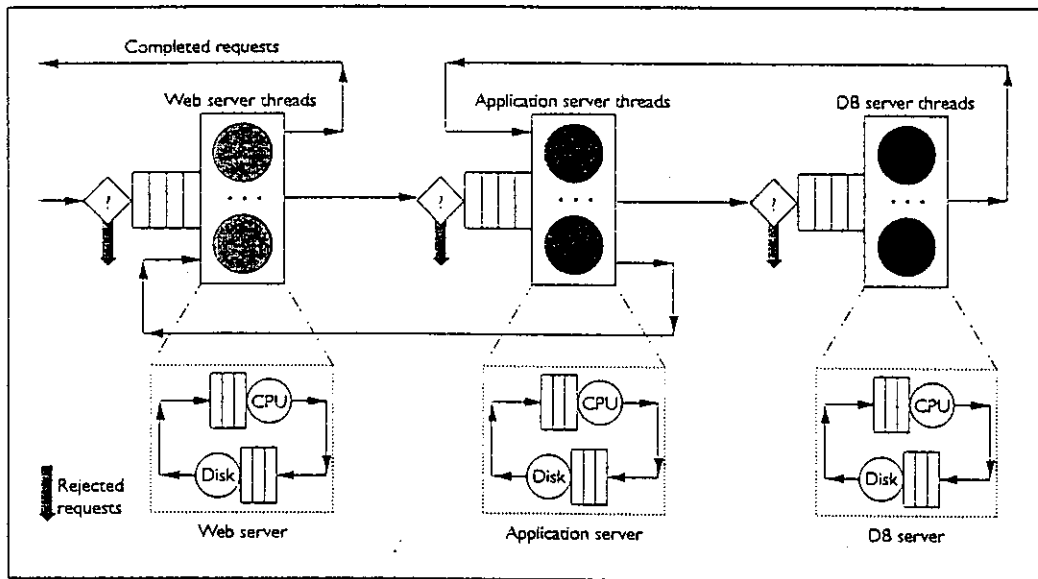


Figure 5. Software and hardware contention in an e-commerce site. The top part of the figure illustrates how requests queue for software resources, such as threads at the Web, application, and database servers. The bottom part of the figure shows that, as threads execute, they use or queue for physical resources such as CPUs and disks.

2. D.A. Menascé, "Load Testing of Web Sites," *IEEE Internet Computing*, vol. 6, no. 4, July/August 2002, pp. 70-74.
3. D.A. Menascé and V.A.F. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*, Prentice Hall, 2002.
4. D.A. Menascé and V.A.F. Almeida, *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*, Prentice Hall, 2000.
5. D.A. Menascé, R. Dodge, and D. Barbara, "Preserving QoS of E-Commerce Sites through Self-Tuning: A Performance Model Approach," *Proc. 2001 ACM Conf. E-Commerce*, ACM Press, 2001, pp. 224-234.
6. V.J. Rayward-Smith, I.H. Osman, and C.R. Reeves, eds., *Modern Heuristic Search Methods*, John Wiley & Sons, 1996.
7. D.A. Menascé, "Two-Level Iterative Queuing Modeling of Software Contention," *Proc. 10th IEEE Int'l Symp. Model- ing, Analysis, and Simulation of Computer and Telecommunication Systems (Mascots 2002)*, IEEE Press, 2002, pp. 267-275.
8. D.A. Menascé, "TPC-W: A Benchmark for E-commerce," *IEEE Internet Computing*, vol. 6, no. 3, May/June 2002, pp. 83-87.

Daniel Menascé is a professor of computer science, the co-director of the E-Center for E-Business, and the director of the MS in E-Commerce program at George Mason University. He received a PhD in computer science from UCLA. His published books include *Capacity Planning for Web Services* and *Scaling for E-Business* (Prentice Hall). He is a fellow of the ACM and a recipient of the A.A. Michelson Award from the Computer Measurement Group.

**Editorial:** *IEEE Internet Computing* targets the technical and scientific Internet user communities as well as designers and developers of Internet-based applications and enabling technologies. Instructions to authors are at <http://computer.org/internet/author.htm>. Articles are peer reviewed for technical merit and copy edited for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society.

**Copyright and reprint permission:** Copyright ©2003 by the Institute of Electrical and Electronics Engineers. All rights reserved. Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of U.S. copyright law for private use of patrons those articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Dr., Danvers, Mass. 01923. For copying, reprint, or republication permission, write to Copyright and Permissions Dept., IEEE Service Center, 445 Hoes Ln., Piscataway, NJ 08855-1331.

**Circulation:** *IEEE Internet Computing* (ISSN 1089-7801) is published bimonthly by the IEEE Computer Society. IEEE headquarters: 3 Park Avenue, 17th Floor, New York, NY 10016-5997. IEEE Computer Society headquarters: 1730 Massachusetts Ave., Washington, DC 20036-1903. IEEE Computer Society Publications Office: 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, Calif. 90720; (714) 821-8380; fax (714) 821-4010. Subscription rates: IEEE Computer Society members get the lowest rates and choice of media option — US\$37/30/48 for print/electronic/combo. For information on other prices or to order, go to <http://computer.org/subscribe>. Back issues: \$10 for members, \$20 for nonmembers. Also available on microfiche.

**Postmaster:** Send undelivered copies and address changes to *IEEE Internet Computing*, IEEE Service Center, 445 Hoes Ln., Piscataway, NJ 08855-1331. Periodicals postage paid at New York, N.Y., and at additional mailing offices. Canadian GST #125634188. Canada Post International Publications Mail Product (Canadian Distribution) Sales Agreement #1008870. Printed in USA.