Reputation Management for Collaborative Content Distribution[*]

Anurag Garg   Roberto Cascella

Dipartimento di Informatica e Telecomunicazioni,
Università di Trento,
Via Sommarive 14, 38050 Povo (TN), Italy.
E-mail: {garo,cascella}@dit.unitn.it

Abstract

We propose a reputation-based trust management system, ROCQ, to reduce inauthentic and corrupted file transfers in end-user collaborative content-distribution systems. Such systems are characterized by the splitting of large files into (optionally encoded) blocks and the simultaneous downloading of several blocks from different nodes to speed up content distribution. All nodes must cooperate and provide correct content for such systems to function smoothly. If malicious nodes are present, they can quickly bring the system to a halt by introducing fake blocks in the network making it impossible to reconstruct the original file.

ROCQ (Reputation, Opinion, Credibility and Quality), uses feedback from past interactions between nodes to create node reputations. This allows detection of malicious nodes in a transaction-based network. In this paper, we study the performance of ROCQ as used in a content distribution network. We find that ROCQ decreases the likelihood of a user receiving fake blocks by upto 16% and hence significantly reduces bandwidth waste.

Keywords: Reputation based trust management, content distribution networks, collaborative networks, end-user collaboration.

1. Introduction

Autonomic content distribution systems need to to be self-managing and self-healing in order to respond autonomically to the users' requests and to changes in the network. These systems are characterized by end-system cooperation (thus also termed collaborative content distribution systems) where content is distributed amongst peers instead of in the standard client-server fashion. The components of such systems must incorporate methods to react in automated ways in order to effectively reconfigure themselves to adapt to new environments. Moreover, the dynamic behavior of these components needs to be monitored constantly. This enables components to propagate information in the network, and at the same time, to respond autonomically to network changes such as node departures and arrivals.

End-system cooperation is the latest step in the evolution of content distribution on the Internet that has progressed from mirrors to proxy caches to server farms to Content Distribution Networks (CDNs). Collaborative end-user content-distribution systems are fast becoming the new paradigm for content delivery as they are cost-effective, inherently scalable and more resilient to network and equipment failures. These systems are useful when several users are interested in downloading the same file from a single server. They exploit the fact that while server bandwidth is limited, idle bandwidth is available between the users themselves. The target file can be split into blocks which can be downloaded simulatenously by different end-users. The end-users can then collaborate and share these blocks with each other to reconstruct the original file. This reduces the load on the server and makes more efficient use of available bandwidth, reducing the overall download time. Moreover, as more nodes join the network, more bandwidth becomes available between end-users making the system inherently scalable. Autonomic collaborative systems also successfully handle flash crowds caused by documents that suddenly become very popular putting the origin server under strain.

However, making end-systems service providers in addition to being service consumers increases reliance on machines that may be less secure than dedicated servers or proxy caches that formed the backbone of earlier systems. Even worse, some of these systems may be controlled by malicious users that wish to disrupt the content-distribution network. Since the participation of nodes in a collaborative network is strongly dependent on whether, and how much, they benefit from it, ensuring compliance of individ-

[*] This work was supported by the Provincia Autonoma di Trento through the WILMA Project and by the European Commission through the E-Next initiative.

ual nodes to the overall goals becomes critical to the success of the system. Two major concerns that have been the subject of research are free-riding, where users download files without "giving back" to the network, and malicious users who introduce fake or mislabeled blocks in the network making reconstruction of the original file difficult if not impossible. In this paper, we address the second of these concerns.

If tampered or mislabeled content is downloaded, the receiving peer must attempt to retrieve the content again. If the fake blocks can be identified, only those blocks need to be downloaded again. However, if there is no mechanism to check the integrity of individual parts of the content, as is often the case when the blocks are encoded, the entire file has to be downloaded again. This magnifies the impact of malicious behavior on the network. Therefore, the behavior of the nodes participating in the content distribution must be monitored carefully.

This paper explores ideas of reputation management that have been developed in other contexts and applies them for soft management in autonomic and collaborative content distribution systems. We examine the ROCQ scheme, (Reputation, Opinion, Credibility and Quality), proposed in [10], a feedback-based trust management system that uses opinions from past interactions to measure the trustworthiness of a peer. We aim at incorporating methods to sustain self-optimization of the resources by limiting corrupted downloads and to provide a minimal degree of protection of the shared content. In this work, we extend the ROCQ model to function in a collaborative content distribution system. We simulate both the scenarios we mentioned above, i.e., when the node(s) providing the fake blocks can be identified and when they cannot be identified. We then run detailed simulations of the latter case – which we believe is a more accurate representation of systems currently in place – and measure the impact of several variables on the performance of ROCQ.

We start with an overview of related work in the next section. The system model is summarized in Section 3 along with a description of the changes made to ROCQ for adapting it to a content-distribution system. In Section 4 we present our experimental methodology and results. We conclude in Section 5, and discuss open problems.

2. Related Work

Several techniques for collaborative content distribution have been proposed recently. Initial proposals focused on extending overlay multicast architectures and constructing multiple parallel overlay trees such as in SplitStream [5]. Mesh architectures provide an alternative to tree-based systems allowing end-nodes to benefit from additional connections. The most popular of these is BitTorrent [6] which is a peer-to-peer application to enable fast downloading of popular files. Because there is no predetermined path in meshes and content can cycle within the mesh indefinitely, nodes need to coordinate download decisions. BitTorrent's solution is to download random blocks in the initial phase of a download and then download the block that is rarest in a node's neighborhood. While this approach is simple, it is not globally optimal.

Alternative schemes have proposed block encoding as a method to improve the efficiency of content propagation by reducing the need for node coordination. These schemes allow a large number of distinct blocks to be generated from a file. The advantage of coding is that only a small subset of the blocks has to be downloaded to reconstruct the original file. However, care must be taken to ensure that the blocks do not overlap too much. Two popular encoding approaches are Erasure Codes [3, 4] and Network Coding [2, 11]. In the former, new codes can only be generated by the server, whereas in the latter, any node can generate new codes for a file based on a linear combination of all received blocks for that file present at that node.

However, encoding blocks makes the problem of verifying block integrity more difficult. A simple checksum is no longer sufficient to check whether a block has been tampered with in transit. The problem is compounded when rateless codes are used, as in this case, the total number of codes that can be generated is very large. Krohn et al. [13] have proposed a homomorphic collision-resistant hash function as a solution to this problem. Thier solution requires the source node to generate a hash value for the entire file and other nodes in the network to download this hash to be able to verify block integrity on-the-fly. They show that this homomorphic hash function is independent of the encoding rate allowing it to be used with rateless codes. Nevertheless, the problem remains unsolved in the case of network coding where any node in the network can generate a new code. Therefore, it is not possible to generate hash values for all the possible codes *a priori*.

The techniques to protect content focus on checking the integrity of individual blocks. If a block is corrupted, it can be identified and all the downloaded blocks for that file do not have to be discarded and downloaded again. Using reputation systems, we can identify malicious nodes and thus prevent downloading of the corrupted block itself saving even more bandwidth. Moreover, the reputation approach works even in the case of network coding.

Considerable research has been done on reputation systems that motivate peers to collaborate and to behave honestly. Initial efforts at trust management in electronic communities were based on centralized trust databases that stored the ratings provided by the users [8, 16]. In the context of peer-to-peer systems, Aberer et al. [1] introduced a reputation scheme that used a decentralized storage system, *P-Grid*. After a transaction, a peer could file "complaints" against other transaction participants if those peers behaved maliciously. The complaints were stored at other peers called agents. Many other transaction-based reputation systems for peer-to-peer networks have emerged since then [7, 10, 12, 15].

3. System Model

The ROCQ reputation management scheme is independent of the underlying co-operative content distribution mechanism. ROCQ was initially proposed in the context of virtual communities where all transactions are one-to-one. We now modify it to be used with content distribution scenarios where a single file is encoded into several blocks. We give a brief description of the algorithm here. Further details can be found in [9, 10].

ROCQ computes global **reputation** values for peers on the basis of first-hand **opinions** of transactions provided by the participants. These opinions are weighted according to the credibility of the reporting peer and the attached quality value. The **credibility** of a peer signifies its trustworthiness in the reputation system. It is a measure of the agreement between the opinion of the peer and that of other reporting peers. The **quality** value allows a peer to reduce the impact of its opinion on the reputation of the target peer. This is useful when the transaction involved was less important or when the reporting peer is not sure of its opinion. For the purpose of rating, it is assumed that each individual peer is identified by a unique ID. The global reputation values are stored in a **decentralized** fashion using multiple *score managers* [9, 10] for each individual peer. Before entering a transaction, a peer retrieves the reputation value of its potential partner from the score managers in order to decide if it should go ahead with the transaction.

The basic algorithm is unchanged in content distribution networks except that now a transaction fails or succeeds and opinions are formed only after **all the blocks** required for reconstructing a file have been downloaded. If the file cannot be correctly reconstructed after receiving the required number of blocks, one or more of the blocks are corrupted. If blocks can be individually checked only the senders of bad blocks will get a negative rating. If they cannot be checked all senders will receive a negative rating.

4. Experimental Evaluation

In this section we study the performance of ROCQ in reducing inauthentic downloads and in detecting malicious peers in a collaborative content distribution network. We use FreePastry [14], an open-source implementation of Pastry that is written in Java, to locate and route messages over this P2P network. ROCQ is implemented as an application that runs on top of individual Pastry nodes. The parameters' settings and performance and decision metrics that we use to evaluate the results obtained from the simulations are listed below and summarized in Table 1.

Parameters that are fixed for all experiments

# nodes	1000
# content transactions	5000
# score managers	6
# experiments run	10
type of node maliciousness	reports and file
decision metric	reputation plus local opinion
type of decision	deterministic
trust threshold	0.5
network topology	random

Parameters that vary in the experiments

# blocks	$k \in \{1, 2, 4, 8, 12\}$
% malicious peers	$frac. \in \{1, 2, 4, 8, 12, 16\}$

Table 1. Experimental parameters

Number of nodes and interactions. In our experiments, we simulate a network of **1000** peers with **5000** content transactions taking place in each simulation run. Note that each transaction involves downloading of k blocks, where k lies in the set $\{1, 2, 4, 8, 12\}$. The default number of score managers storing reputation ratings for each peer is 6. Each experiment is performed 10 times and the average of the results is plotted, along with error bars signifying the *standard error*.

Performance metric. To evaluate the performance of ROCQ, we calculate the proportion of sucessful file transfers as a proportion of the total number of content transactions. We also compute the number of correct decisions made (i.e. interactions with good peers that proceeded plus interactions with malicious peers that were avoided) as a proportion of the total number of decisions made.

Types of maliciousness. We assume that peers behave maliciously in both the content distribution system and in the reputation system. A malicious peer will send a corrupt block in response to a request. It will also give an incorrect opinion (or reputation) value to another peer in its capacity as a transaction partner (or score manager). Hence, if O (R) is the actual opinion (reputation) value, the value that is sent is $(1-O)$ ($(1-R)$).

Decision metric. A peer decides whether to interact with another peer based on a combination of the reputation value obtained from the score managers and of the local opinion value that a peer forms based on the first-hand interactions. These two values are averaged and the decision is taken based on the resulting score. We use a threshold of 0.5 and an interaction takes place only when the score exceeds this value. If a peer does not have a local opinion of the behavior of the correspondent peer (thus, they have never interacted), only the reputation value is used. When there is no reputation information available for the correspondent peer, the interaction takes place but this is counted as initial interaction.

Selection of the peers. The simulator is round-based where each round corresponds to the **complete** transfer of k blocks of a file to a peer, where k lies in the set $\{1, 2, 4, 8, 12\}$. At the start of each round we pick a peer at random (source) which then requests blocks from k peers in the network (targets). The same peer can not be picked as a target more than once in the same round. The source checks the trust values of each target and rejects targets that have a reputation value below the threshold. A replacement target is found for each rejected target till we find k targets with sufficiently high trust values.

4.1 Comparison with the No Reputation Management Case

In this experiment we compare the performance of the ROCQ scheme in the content distribution network with the case when no reputation management scheme is in effect. We evaluate the performance of ROCQ both when it is possible to identify the fake block and when it is not possible to do so.

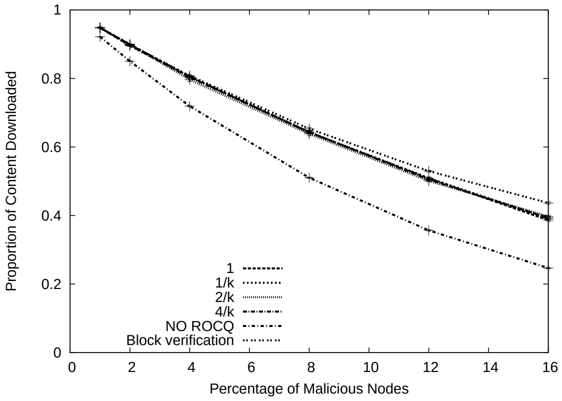

Figure 1. Comparison of the RQC Scheme with No Reputation Management Case ($k = 8$)

When it is not possible to identify the fake block(s) in a collaborative download, the content is assumed to be successfully downloaded only if all the k blocks are uncorrupted. If even one of the blocks is corrupt, the entire download has to be repeated. Since all the peers that sent a block are potentially malicious, the recipient peer files a negative trust report for each of these peers. On the other hand, if the file is successfully reassembled, all the peers receive a positive trust rating.

Because one fake block can result in several negative reports, we also experiment with reducing the weight of negative opinions. Positive opinions are always given a weight of 1, while negative opinions are given the weight: 1, $\frac{1}{k}$, $\frac{2}{k}$ and $\frac{4}{k}$ (where k is the number of blocks and the peers involved in the transaction).

We simulate the source requesting 8 blocks (thus $k = 8$) from different peers in the network. Figure 1 shows that ROCQ results in up to 16% more correct downloads than when no reputation management scheme is used. We also see that varying the weight given to negative opinions does not have any significant impact on the performance of ROCQ.

ROCQ performs well in both cases when individual nodes uploading malicious content can be identified and when they cannot be detected. This is because in ROCQ a peer decides to interact with another peer based on its reputation value. As a result, malicious peers may be easily detected before initiating a transaction. Therefore, a fine-grained detection of malicious peers only slightly improves the performance of ROCQ.

In the remaining experiments, we assume that individual bad blocks cannot be identified as this is the worst case scenario.

4.2 Number of Required Blocks

In this experiment we measure the impact of the number of blocks required to reconstruct a file on ROCQ's performance in content distribution. Figures 2 and 3 show that the proportion of content successfully downloaded and the percentage of correct decisions decrease as we need more blocks to reconstruct a file. We plot different lines for different percentages of malicious peers in the network (ranging from 1% to 16%).

While a larger number of blocks implies smaller block size and greater collaboration resulting in better performance, it also increases the risk of exposure to malicious nodes. Our experiments do not attempt to measure this trade-off between performance and exposure to maliciousness but they do indicate that as the proportion of malicious nodes in the network increases, the optimal number of blocks that should be needed to reconstruct a file decreases.

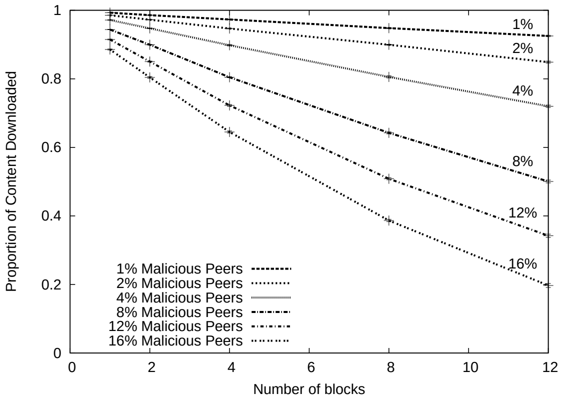

Figure 2. Impact of Number of Required Blocks on Proportion of Successful Downloads

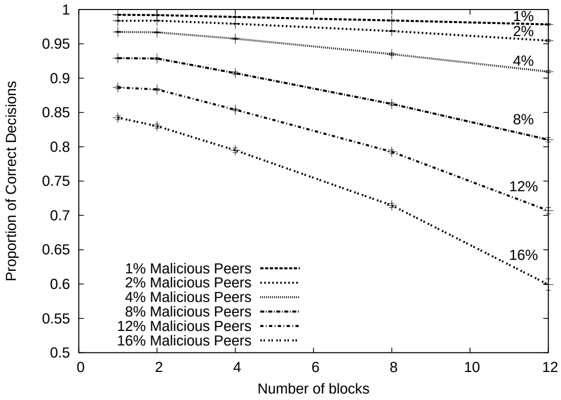

Figure 3. Impact of Number of Required Blocks on Proportion of Correct Decisions

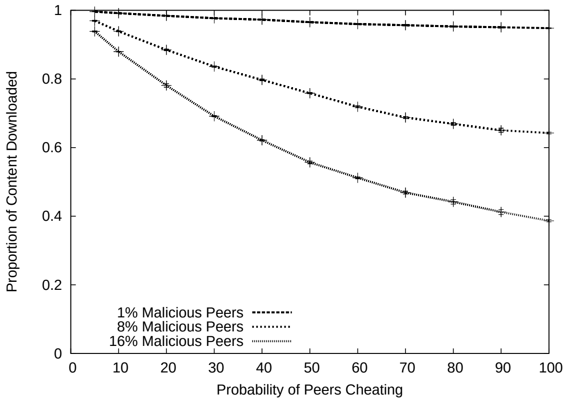

Figure 4. Proportion of Successful Downloads vs. Chance of Peer Acting Maliciously (with $k = 8$)

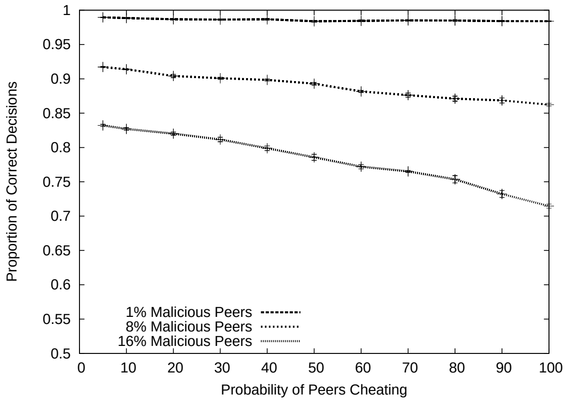

Figure 5. Proportion of Correct Decisions vs. Chance of Peer Acting Maliciously (with $k = 8$)

4.3. Occasional Maliciousness

When peers act maliciously in a consistent fashion, it should be possible to identify them relatively quickly. However, if peers choose to act maliciously only some of the time, it may be more diffcult to identify such **occasional cheaters**.

In Figures 4 and 5 we examine the case when the malicious peers cheat only some of the time. We simulate the source requesting 8 blocks from different peers in the network (thus, $k = 8$). The three curves correspond to a total of 1%, 8% and 16% of the nodes being malicious. Figure 5 shows that the proportion of correct decisions is only slightly affected by the probability of a node cheating. However, Figure 4 shows that proportion of successful downloads decreases much more sharply as the percentage of time nodes cheat increases. Since the proportion of correct decisions has decreased only slightly, we can see that ROCQ has identified malicious nodes with the same relative success. The reduction in the proportion of correct downloads can be attributed simply to the larger proportion of malicious transactions. This also demonstrates how in a collaborative content distribution network, a little bit of malicious behavior can have a disproportionate impact on the network.

5. Conclusions

In this paper we present experimental results after applying an extended model of ROCQ for reputation management in collaborative content distribution networks. Unlike other reputation management schemes, our approach targets collaborative content distribution through splitting files into blocks and uploading them to different peers followed by the exchange of these blocks amongst peers.

Our solution to the problem of malicious corruption of blocks is to allow a peer to select its transaction partners based on the trust he/she places on them. A collaborative content distribution system is particularly susceptible to *pollution attacks* since a single bad block can render the entire download worthless. With ROCQ, a large percentage of malicious nodes are identified and corrupt block transfers are prevented.

Our experimental results demonstrate the advantage of using ROCQ in a collaborative content distribution network as opposed to not using a reputation management scheme. We show that ROCQ improves the proportion of correct downloads by up to 16% and performs well both when the sender of the corrupt blocks can be identified and when they cannot be identified. In addition, we evaluate the impact of the number of blocks and the percentage of malicious peers on the performance of ROCQ. As the number of blocks required increases, the probability that the source downloads the content successfully decreases. Hence networks with a large number of malicious nodes should keep the number of blocks required to reconstruct a file as low as possible.

An immediate applicability of our work is to content networks relying on network coding. With network coding, blocks are re-coded at each intermediate node, and block verification cannot be done by using a predetermined hash of the block. Hence, the scheme proposed by Krohn et al. in [13] will probably be ineffective.

We are currently in the process of enhancing our reputation scheme for managing a system where network coding is used for block encoding. We are also investigating other methods for improving the performance of our scheme. These include strategies such as downloading more than the minimum required number of blocks. We will consider downloading $k+1$ blocks when k blocks are required. Obviously, this strategy cannot be extended to much higher numbers of *spare blocks* as the number of possible recombinations increases very rapidly.

References

[1] K. Aberer and Z. Despotovic. ”Managing Trust in a Peer-2-Peer Information System”. In *The Tenth International Conference on Information and Knowledge Management CIKM*, pages 310–317, Atlanta, Georgia, USA, 2001.

[2] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. ”Network Information Flow”. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[3] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost. ”Informed Content Delivery Across Adaptive Overlay Networks”. *IEEE/ACM Transactions on Networking*, 12(5):767–780, October 2004.

[4] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. ”A Digital Fountain Approach to Reliable Distribution of Bulk Data”. In *SIGCOMM*, pages 56–67, 1998.

[5] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. ”SplitStream: High-bandwidth content distribution in cooperative environments”. In *The 2nd International Workshop on Peer-to-Peer Systems (IPTPS ’03)*, Berkeley, CA, 2003.

[6] B. Cohen. ”Incentives Build Robustness in BitTorrent”. In *1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, June 5-6 2003.

[7] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. ”Managing and Sharing Servents’ Reputations in P2P Systems”. *IEEE Transactions on Data and Knowledge Engineering*, 15(4):840–854, July/August 2003.

[8] C. Dellarocas. ”Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior”. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 150–157. ACM Press, 2000.

[9] A. Garg and R. Battiti. The Reputation, Opinion, Credibility and Quality (ROCQ) scheme. Technical Report TR04-104, Department of informatics and Communication Technologies, University of Trento, December 2004.

[10] A. Garg, R. Battiti, and G. Costanzi. ”Dynamic Self-management of Autonomic Systems: The Reputation, Quality and Credibility (RQC) scheme”. In *The 1st IFIP TC6 WG6.6 International Workshop on Autonomic Communication (WAC 2004)*, Berlin, Germany, October 2004.

[11] C. Gkantsidis and P. Rodriguez. ”Network Coding for Large Scale Content Distribution”. To appear in *IEEE INFOCOM 2005*, Miami, FL, USA, March 13-17 2005.

[12] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. ”The Eigentrust algorithm for reputation management in P2P networks”. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 640–651. ACM Press, 2003.

[13] M. N. Krohn, M. J. Freedman, and D. Mazires. ”On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution”. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 9-12 2004.

[14] A. Rowstron and P. Druschel. ”Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems”. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[15] L. Xiong and L. Liu. ”PeerTrust: Supporting Reputation-Based Trust in Peer-to-Peer Communities”. *IEEE Transactions on Data and Knowledge Engineering, Special Issue on Peer-to-Peer Based Data Management*, 16(7):843–857, July 2004.

[16] G. Zacharia, A. Moukas, and P. Maes. ”Collaborative Reputation Mechanisms in Electronic Marketplaces”. In *Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8*. IEEE Computer Society, 1999.