Flow-Net Methodology for Accountability in Wireless Networks

Yang Xiao, University of Alabama

Abstract

Accountability implies that any entity should be held responsible for its own specific action or behavior so that the entity is part of larger chains of accountability. One of the goals of accountability is that once an event has transpired, the events that took place are traceable so that the causes can be determined afterward. The poor accountability provided by today's computers and networks wastes a great deal of money and effort; examples include activities to identify whether a system is under reconnaissance or attack, and the difficulties of distinguishing legitimate emails from phishing attacks. This is due to the simple fact that today's computing and network infrastructure was not built with accountability in mind. In this article we propose a novel methodology called flow-net for accountability. We apply this methodology to media access control and routing layers in wireless networks. We then compare the performance of flow-net with audit log files. This article presents a novel approach for traffic data collection that can also be used for forensics and intrusion detection purposes.

Current trace/logging systems in computers and networks have many drawbacks such as disorganized and inconvenient data, partial activities being recorded, and relationships among events missing, which make it impossible to trace some information if events at different layers are involved. Therefore, accountability is not possible. In this article, we propose a flow-net methodology for better accountability and logging

Accountability implies that any entity (e.g., person, software, hardware, account, process, flow) should be held responsible for its own specific action or behavior so that the entity is part of larger chains of accountability. One of the goals of accountability is that once an event has transpired (e.g., leaking of secure information or outside attack), the events that took place are traceable so that the causes can be determined afterward. For example, was this a malfunction of software/hardware or a malicious attack? If it is leakage of secure information, we need to know where and how the secure information was disseminated. Accountability is extremely important for military and government systems. For non-military/government systems, some degree of accountability is also needed. People are normally unwilling to see installations of spyware/adware. It will be very useful that, in current Windows-based systems, installed software will be accountable so that actions, such as modifying/replacing software in a system by a malicious Website that exploits software vulnerabilities in operating systems via clicking the Website links via a browser, can be traceable. The poor accountability provided by today's computers and networks wastes a great deal of money and effort (e.g., activities to identify whether a system is under reconnaissance or attack, or difficulties in distinguishing legitimate emails from phishing attacks). This is due to the simple fact that today's computing and network infrastructure was not built with accountability in mind.

Typically, there are two complementary classes of approaches to network security: *prevention-based* and *detection-based* [1]. Prevention-based techniques, such as authentication and encryption, can effectively reduce attacks by ensuring that users conform to predefined security policies. They can keep most illegitimate users from entering the system. However, security research indicates that there are always some weak points in the system that are hard to predict [1]. To solve these problems, detection-based approaches such as virus detection and intrusion detection systems, serving as the second wall of protection, could effectively help identify malicious activities. However, neither prevention-based nor detection-based approaches can track what happens afterward (i.e., traceability).

Accountability plays a crucial role in information assurance systems. There are many fields involved in achieving accountability, including hardware, operating systems, programming languages, software, networking, applications, and security. Most existing research work has focused on e-commerce accountability [2, 3], Web access accountability [4, 5], programming accountability [6, 7], digital rights management [8, 9], audit trail [10, 11], and so on. For example, for Web accountability, it is possible to associate individuals with actions on Web-based objects and services. For programming accountability, specific flows due to particular program invocations can be known and limited to a particular flow policy. For digital rights management, data are packaged for access controls in remote platforms. To provide accountability in information assurance, we may be interested in tracking who has read or written a particular file or a piece of the file, as well as the sources of the file or a piece of the file, and compromised material in the file if there is a malicious insider. Sources of information, such as messages, reports, images, videos, and articles, are particularly useful in

0890-8044/09/$25.00 © 2009 IEEE

accountable systems to identify all the contributors. An audit trail may be needed to record all of these. However, current techniques [2–11] still cannot provide true accountability, as evidenced by the fact that current computer systems and networks have not provided much assistance in tracking actual information flows or enforcing information flow policies. Furthermore, most research efforts in wired/wireless networks focus on prevention-based and detection-based security, whereas less effort has been made regarding accountability.

Instead of pursuing enhancements of either prevention-based or detection-based approaches, we are pursuing dramatic improvement in accountability with different levels by radically redesigning today's computing and network's infrastructure into a more flexible, usable, and trustworthy infrastructure. In this article we first propose a novel methodology called flow-net for accountability. We apply this methodology in the media access control (MAC) and routing layers of wireless networks.

Accountability via Flow-Net Methodology

In this section we propose a novel methodology called flow-net for accountability. The proposed methodology is not limited by the studies presented in this article, but can easily be applied to other systems and networks.

Accountability

We propose two aspects for accountability:

- **Building accountable audit data**: to build tracing files of events: how to build audit data to provide enough material to achieve accountability. Since audit data are huge, the challenge issues include how, where, and what data to save (events). In this aspect we propose a *multilevel, multiresolution* account of events for accountable audit data by using 3D flow-net methodology in the rest of the article.
- **Accountability functions**: using audit data: how to use the collected audit data to achieve accountability.

Building accountable audit data is one foundation of accountability, and must be achieved based on the requirements of using audit data, which is about how to use audit data to achieve traceable functions as well as, for example, how to perform intrusion detection conveniently. We propose several accountability functions, which are summarized as follows.

A Flow-Net Truth-Tracking Methodology — A way to track the truth: once an event has transpired (e.g., leaking of secure information or outside attack), the events that took place are traceable so that the causes can be determined afterward. For example, was this a malfunction of software/hardware or a malicious attack? If it is leakage of secure information, we need to know where and how the secure information was disseminated. This is akin to how policemen solve criminal cases, and it is different from traditional computer forensics since computer forensics aim to solve cases with limited information. However, our proposed research is to study how to build a system (or methods) so efficient, and with enough information, that it makes computer forensics (one of the potential applications of accountability) very trivial.

True Accountable Administration — Since an administrator or a super user has the authority to make changes in anything, even audit files (or log files), achieving accountability for administrators or super users is extremely challenging. One intuitive solution is that other audit files are used for the audit files (i.e., audit files for audit files), but a malicious super user can make changes to both audit files and audit files for audit files so that any trace can be smartly removed. We proposed a scheme with accountable administration (i.e., even an administrator's activities must be accountable) in [12]. Please refer to [12] for the details of our proposed accountable administration.

Motivation of Flow-Net over Audit Log Files

Current audit data in networks such as log files in operating systems, routers, and so on cannot provide accountability. One of the challenging issues is that there is a huge amount of information needed to achieve accountability. It becomes difficult to store such a huge amount of information. Furthermore, when performing accountability functions, it is difficult to find a needle in that haystack. Traditional audit trace/log files have many drawbacks:

- The files are huge and disorganized, so it is very inconvenient to use them (i.e., tracing some).
- Normally only partial activities are recorded.
- They often lose important information about the relationships among events, especially relationships in the same layer and multiple different layers, so it is not possible to trace some information if events at different layers are involved.

Flow-Net Methodology

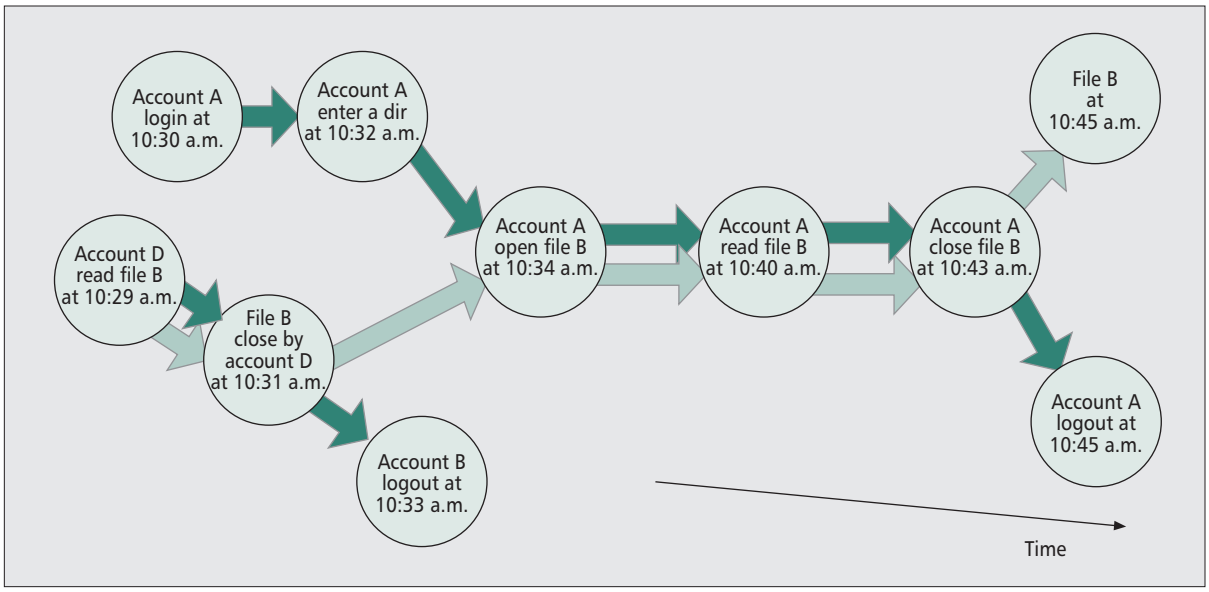
One question is how to trace efficiently. We propose a flow-net methodology with the following features: convenient to use, complete in terms of information, maintaining full relationships among different layers, and consistent over different systems.

The saved information includes some events such as opening/reading a file in an operating system or sending a frame in the MAC layer. These events over time form event-lists. Traditional log files are basically saved as event-lists. Events in the MAC layer include authentication, association, broadcasting, clear channel assessment, transmit a data frame, transmit an acknowledgment (ACK) frame, receive a data frame, receive an ACK frame, request to send (RTS), clear to send (CTS), Collision, polling, fragmentation, defragmentation, reassociation, timeout, retransmission, wake up, sleep, drop a frame, etc.

To illustrate the idea of flow-net, we make an example in an operating system: each entity, such as a file, a user, or a process, has a log trace, which is an event-list with time stamps — we call it a flow. An interconnection of two flows at a time means that something happens at that time, involving both entities. For example, user A (event-list Account-A) interconnects with File B (event-list File-B) at 10:34 a.m. when user A opens file B (Fig. 1). In other words, all entities have their event-lists, which we call traces or flows. These flows (event-lists) interconnect with each other over time stamps, shown in Fig. 1. Then all these interconnected flows form a net over time because they interconnect. Therefore, we call it a flow-net.

3D Flow-Net: Multilevel Multiresolution Event Storage for Accountable Audit Data

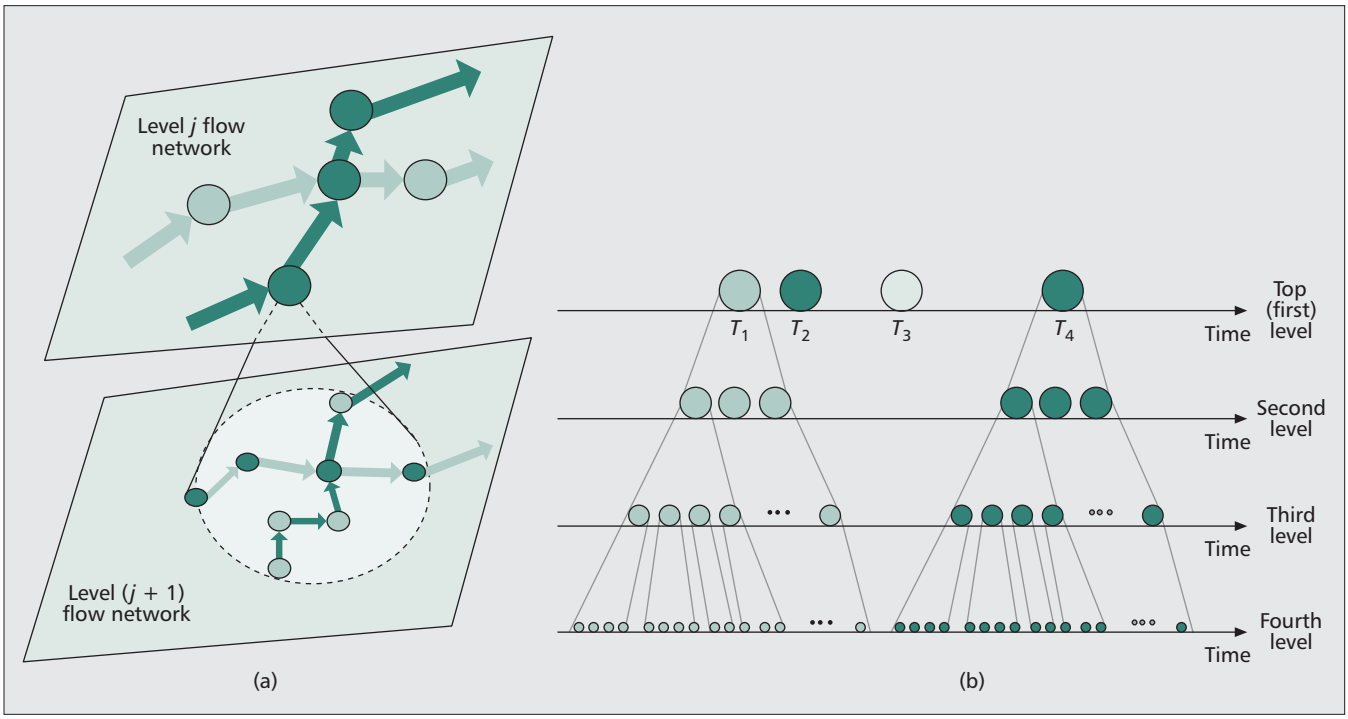
We propose a multilevel and multiresolution approach to storing event lists. Event lists should be built in such a way that they look like a multilevel multiresolution image/video with zooming effects. As an analogy, consider a multiresolution image on a screen. At first we see a large scene of a picture without seeing details of particular parts/areas of the picture. Then, if we want to clearly see one particular area, we choose the area using a mouse and zoom in with



■ Figure 1. *Flow-net methodology.*

a higher resolution, and then we can clearly see the particular area we are interested in with a larger picture of that area and at a higher resolution. We propose to store event lists in a smart way so that we can achieve the same/similar effects of the above mentioned multilevel multiresolution image/video. With such a multilevel multiresolution approach, we can efficiently achieve multiple degrees of accountability and multiple degrees of audit file storage in order to exploit the trade-off of storage vs. accountability. For example, a military application may use a very high level of accountability, whereas a commercial application for civilians may use a very low level of accountability. These levels of accountability are achieved by the resolutions of storing events, shown in Figs. 2a and 2b. The question is how to achieve such a storage data structure.

We propose to use a similar approach of saving multiresolution versions of an image/video to store events. Let us look at an example of monitoring a government building such as an FBI building. The building is divided into rooms, doors, and corridors. The big events are at certain times, and various people enter/leave different portions of the building, such as entering a door or leaving a door (we call these level-one events). Then, within a room, we can still further divide the room into portions such as locations or usage of the rooms (we call these level-two events). Thus, we can build multilevel, multiresolution events with zooming effects for monitoring this government building. In a similar way, we can monitor a computer or network system. In a computer, a room can be memory space, hard disk locations, CPU, and so on. For the MAC layer, a



■ Figure 2. *a) Multilevel and multiresolution; b) multilevel and multiresolution (layers).*

room can be those already associated or authenticated. In other words, accountable audit data will also be stored and accessed at different levels of granularity based on their semantics.

Figures 2a and 2b show the concept of a multilevel, multiresolution approach. In Fig. 2a an upper (lower) layer event is related to multiple lower (higher) layer events. For example, the routing layer event sending a packet will involve multiple events of the MAC layer (send an RTS, receive a CTS, transmit the frame, receive an ACK, etc.). This concept is shown in Fig. 2b. Within each level, as shown in Fig. 1, events form a flow-net. From Fig. 2b, we see that sublayer events related to an event in the top layer form a tree structure. Such a tree structure can be saved as a record in the top layer or using indexing and compression techniques. By comparing this with the multiresolution video example, we can see an upper event, and if we are particularly interested in an event, we can look into details of this event by looking at its related sublayer events (i.e., the subtree of the event). In other words, a higher layer subtree is stored into a tree structure of a lower layer node. Such a data structure is a real-time 3D multilevel flow-net, as introduced earlier.

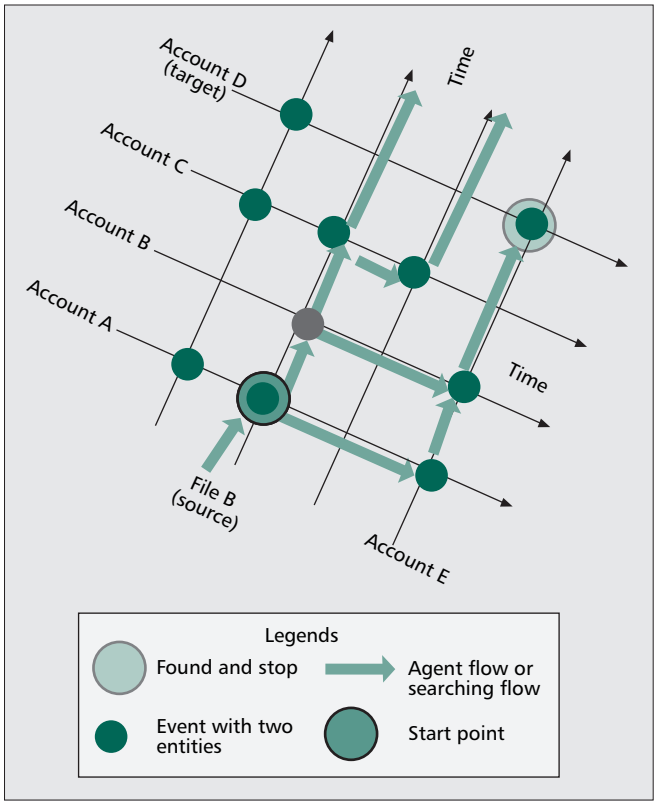
Flow-Net Truth Finding Methodology

Flow-net truth tracking methodology is a way to track the truth. Once an event has transpired (leaking of secure information or an outside attack), the event that took place is traceable so that the causes can be determined afterward. For example, in a computer or network system, if at some point the fact that the content of a secret file is being leaked has already been known, to figure out the reasons for the leak, we can search partial or entire log files to find out direct or indirect accesses to the file. Since a user who accessed the secret before may send messages containing the secret to other users (the secret is leaked due to indirect accesses) via packets in a computer network, or via pipe/FIFO/message queue in a computer system, finding the reasons for the leak is not a trivial task [14]. Using audit log files, the relationships among different layers may be lost so that tracing is sometimes not possible, and it is difficult to search in audit log files. In the proposed flow-net truth finding methodology, tracing the truth becomes much easier. For example (Fig. 3), assume that we want to find the truth of whether or not the information in File B was possibly leaked to Account D between 1 a.m., July 1, 2007, and 1 a.m., July 2, 2007 (recorded system time, not search time). Figure 3 shows that the proposed methods start from the flow of File B and go through the flow-net along the time to search whether it can end up with an event related to Account D within that 24-hour time period (note that an event has time as a field). The algorithm starts at File B's flow and searches the flow-net along the progressing time. The search process is exactly a tree-searching algorithm, which can be depth-first or breadth-first. The worst, best, and average search times can be obtained easily.

The results of the truth searching in Fig. 3 can be:

- There are two possible paths to have such a leak.
- The best case needs three steps to find a possible path.

For example, in one path, Account A accesses File B, and it may be possible to leak the information of File B to Account D. On the other hand, a simple example of the truth finding in Fig. 3 would be very inconvenient to achieve in regular sequential log files. Although Fig. 3 shows an example of 2D flow-net, the above scheme can be used in 3D flow-net, that is, the multilevel flow-net.



■ Figure 3 *Flow-net truth finding (goal: to find whether or how the information in File B was leaked to Account D; results: 1) there are two possible paths to have such a leak; 2) the best case needs three steps to find a possible path).*

Designs of Flow-Net in Networking

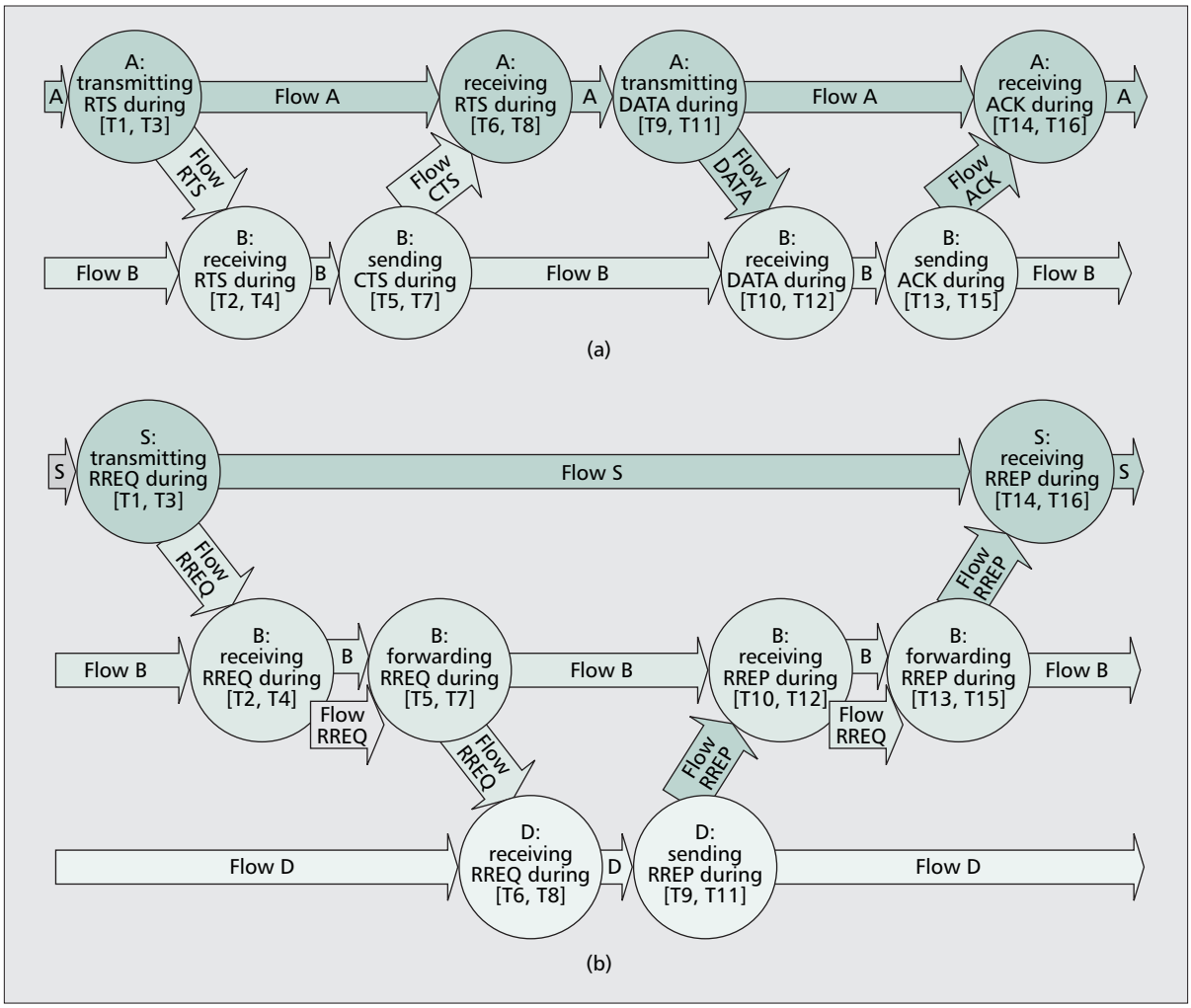
The proposed flow-net methodology has many potential applications. In networking, we realize that frames/packets are the common media to communicate among stations so that transmitted/received frames/packets tie stations together. Furthermore, in networking, normally two or more stations are involved. Entities having flows in networking include stations, packets, and frames, among others.

MAC Level Flow-Net

We consider IEEE 802.11 MAC as an example here. Figure 4a shows a flow-net for RTS-CTS exchanges between stations A and B. To conveniently represent events of starting and ending transmitting/receiving frames/packets, we use duration to simply represent two events as one; for example, in Fig. 4a an event of station A transmitting an RTS during [T1, T3] is a simplified version of two events of station A's starting transmitting the RTS at time T1, and ending this transmission at time T3. This simplifies the flow-net when applied to networking. Figure 4a has six total entities: station A, station B, RTS frame, CTS frame, a DATA frame, and an ACK frame. Event lists of entities are called flows. Their corresponding flows are flow A, flow B, flow RTS, flow CTS, flow DATA, and flow ACK.

Routing Level Flow-Net

One example of a routing protocol for ad hoc networks or mesh networks is Dynamic Source Routing (DSR). In DSR, when a source station S sends a data packet to a destination station D, the entire route is included in the packet header; intermediate nodes use the source route embedded in the packet's header to determine to whom the packet should be forwarded; and different packets may have different routes, even when they have the



■ Figure 4. *a) Flow-net for RTS-CTS exchanges between stations A and B; b) a flow-net for stations S, B, and D for RREQ and RREP.*

same source and destination. Route discovery includes Route Request (RREQ) and Route Reply (RREP). Route maintenance includes Route Error (RERR). Figure 4b shows a flow-net for RREQ and RREP among the source node S, an intermediate node B, and the destination node D.

Interactions of Flow-Nets of Routing and MAC Layers: Multi-level

We use an example to illustrate how to apply 3D flow-net to multilevel multiresolution tracking of events. Figure 5a shows the interaction of flow-nets of the MAC and routing layers for route discovery of RREQ and RREP.

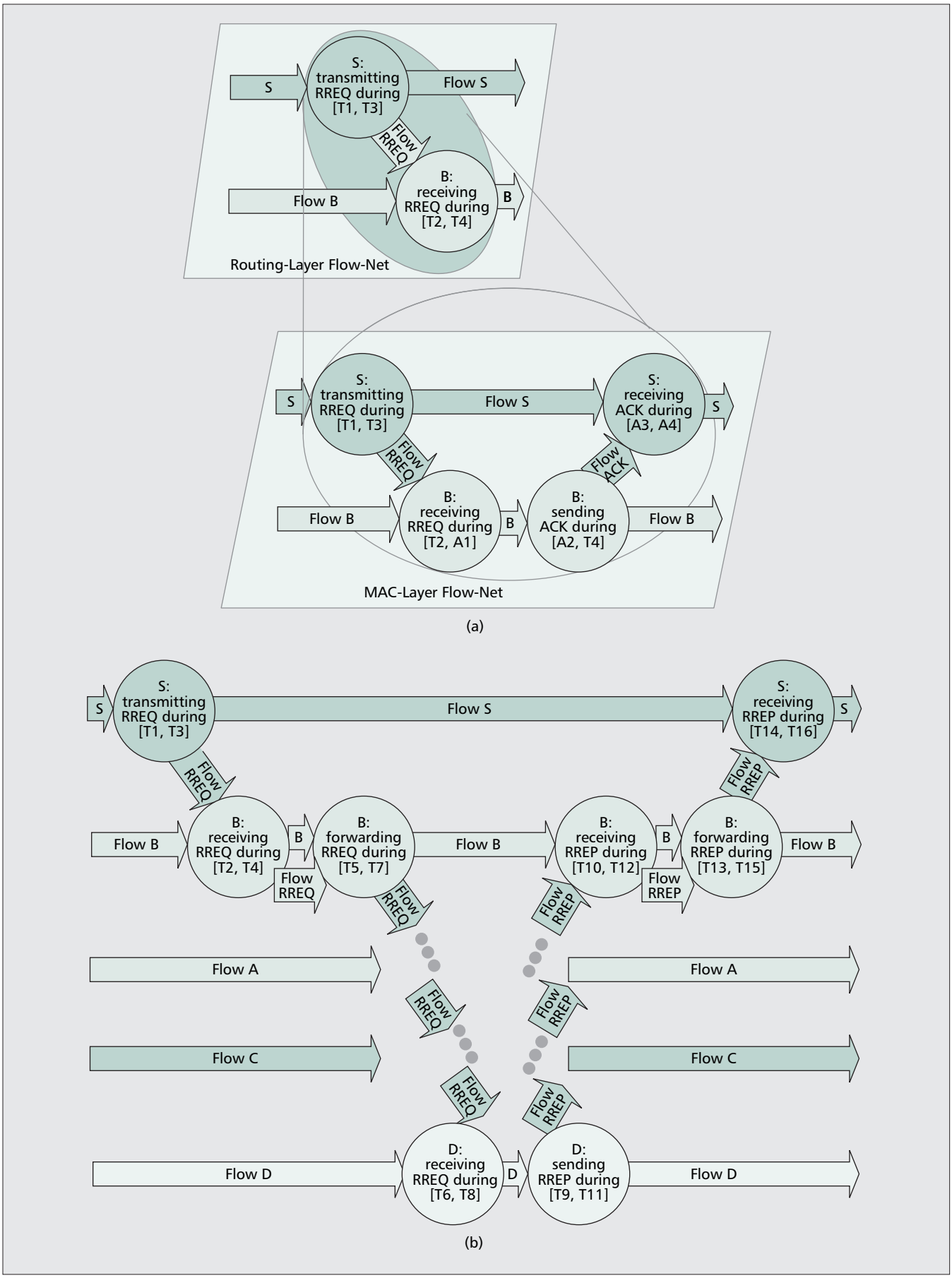
Distributed and Collaborative Flow-Net

A flow-net is built just from a viewpoint of a station or a viewpoint of one-hop. To achieve a larger viewpoint of flow-net, multiple nodes should collaborate so that distributed and collaborative flow-net can be built. Each node has some local flow-nets which present local views of the global picture, but no one has the global view of the system. These distributed views may be exchanged among authenticated nodes to track some events and, if needed, base those distributed and collaborative algorithms. For example, DSR needs multiple nodes to exchange their viewpoints of flow-nets to achieve a larger view. Figure 5b shows a flow-net built in a distributed and collaborative way by multihop nodes for the process of route recovery in DSR.

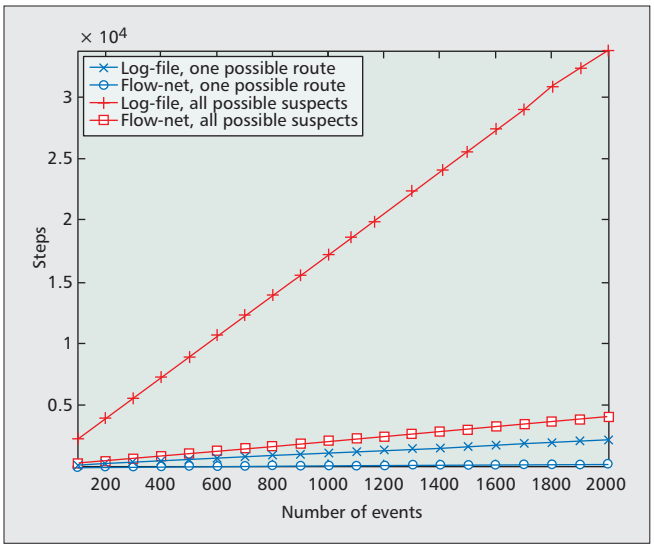
Overhead Reduction

Flow-net may introduce overhead to the system. There is a trade-off between accountability and system performance. Therefore, we propose the following mechanisms for overhead reduction for accountability: multiresolution, multilevel flow-net and flow-net compression.

The size of events can vary. For example, the routing layer can generate high-level events, each of which may cause many low-level MAC layer events. The smaller the events, the larger the related flow-net overhead will be. However, the more detailed the information will be that can be preserved through flow-net. Different systems may have different requirements on how much detail the logging information can have. An accountable system can be coarse- or fine-grained. In other words, the system can have different resolution on the detail of accountable information. Thus, choosing appropriate granularity is very important to avoid introducing unnecessary overhead. As explained earlier, the proposed flow-net methodology is multiresolution and multilevel. In the above, as shown in Fig. 2, we already show two levels of flow-net granularity. We can only use a higher layer (routing layer) flow-net and turn off the lower layer (the MAC layer) to reduce performance overhead. However, there is always a trade-off. We include the detailed trade-off study as well as compression of flow-net in [13].



■ Figure 5. *a) A 3D flow-net for RREQ and RREP at the MAC and routing layers; b) a distributed and collaborative flow-net for multi-hop nodes.*



■ Figure 6. *Complexity comparisons for finding one possible route of information leakage with the depth-first based searching algorithm and of finding all possible suspects concerning an information leakage with the depth-first based searching algorithm.*

Flow-Net Performance Experiments

We implemented the flow-net method as well as the truth finding methodology in wireless LANs. Furthermore, we conducted performance experiments for studying the performance of flow-net and comparing it to log-file. Due to the space and figure limitations of this magazine, we will present our implementation and performance experiments in a future paper [13]. Here we just present one result. Our programs actually create a flow-net and a log-file and run searching algorithms. At this point, a log-file is simply a linked list of logged events ordered by time sequence. Searches start at an event where a message/file that contained leaked information (secret) was first listened to/accessed by some station/user and finish when arriving at an event where information leakage was revealed, similar to our previous work in [14]. Algorithms count the numbers of steps taken. In other words, experiments count how many linked events are visited and the cost of retrieving a current event from time ordered logged events. Execution of searching (counting) was conducted 1000 times, and from the sum of 1000 results, the average was calculated. We considered two problems in Fig. 3:

- To find a possible route/trace on which a secret flows into a particular resource
- To find all the suspects (e.g., users or entities) who possibly directly or indirectly accessed the secret in the past or in the past time duration T

A user indirectly accessed the secret if it contacted another user or entity that directly or indirectly accessed the secret [14].

We measure searching times of finding a possible route for information leakage shown in Fig. 6 with a depth-first searching algorithm. Figure 6 shows that both flow-net and log-file show linear transitions, but the depth-first searching algorithm log-file grows faster than that for flow-net in the figure, and actually log-file involves around 10 times more searching complexity than flow-net (e.g., 1144 to 111 at 1000 events and 2212 to 211 for 2000 events).

At this time, an auxiliary stack is used to store every suspect concerning an information leak. The searches basically traverse the entire binary tree until they finally arrive at the right-most leaf of the binary tree. Note that this is not always the last event in the time sequence. Unlike finding a possible route connecting information leakage, since the searching algorithms inevitably traverse the entire binary tree, their complexity is poorer than those of the previous cases. Therefore, even with the use of flow-net, shown in Fig. 6, it costs more than 4000 at the base value of 2000, whereas it costs 200 to find a possible route. However, compared to log-file, a rate of the complexity rise in flow-net is very efficient because it avoids auxiliary searches for events from the original time sequential record. This avoidance comes from the fact that there are links to associate with related events for a binary tree.

Conclusions and Future Research Directions

This article addresses a very important issue: accountability. A flow-net methodology was proposed for accountability, and it has been applied to MAC and routing layers in wireless networks. Measurement performance of flow-net and audit log file are compared. Furthermore, we propose a flow-net truth finding methodology, and distributed and collaborative flow-net. Via implementing the flow-net, we found that it is a very good method to record details of events. However, flow-net may introduce overhead. To reduce the overhead, we propose two mechanisms: multiresolution, multilevel flow-net and flow-net compression. The proposed novel approach to traffic data collection, the flow-net methodology, can also be used for forensics and intrusion detection purposes since the flow-net methodology adds more features, such as a systematic approach, more events with multiple levels, and a detailed relationship among events.

We further observe that the flow-net methodology has a trade-off of somewhat more construction time than the traditional logging approach [13]. Due to limited space and limited figures, many concepts, detailed designs, detailed evaluations, and implementation of the proposed methods are not included in this article, and readers are urged to read [13] for more details. The implementation of the proposed work is based on our previous work in [15].

Future research directions include optimization of flow-net, trade-off analysis of performance and accountability, forensics analysis based on the flow-net methodology, end-to-end accountability, and intrusion detection based on the flow-net methodology. In the future we will also study how the logging should be done to prevent resource depletion, how much disk space is required for a normal station, and what can be done to compress/reduce the amount of storage.

Acknowledgment

This work is supported in part by the U.S. National Science Foundation (NSF) under grant numbers CNS-0737325, CNS-0716211, and CCF-0829827. The author would like to thank his Ph.D. student, Daisuke Takahashi, for providing a simulation result (Fig. 6) in this article.

References

[1] B. Sun *et al.*, "Enhancing Security using Mobility-Based Anomaly Detection in Cellular Mobile Networks," *IEEE Trans. Vehic. Tech.*, vol. 55, no. 4, July 2006, pp.1385–96.
[2] R. Kailar, "Accountability in Electronic Commerce Protocols," *IEEE Trans. Software Engineering*, vol. 22, no. 5, May 1996, pp. 313–28.
[3] S. Kungpisdan and Y. Permpoontanalarp, "Practical Reasoning about Accountability in Electronic Commerce Protocols," *Proc. 4th ICISC '01*, vol. 2288, Dec. 6–7, 2001, pp. 268–84.
[4] A. L. Rosenberg, "Accountable Web-Computing," *IEEE Trans. Parallel and Distrib. Sys.*, vol. 14, no. 2, Feb. 2003, pp. 97–106.
[5] A.R. Butt *et al.*, "Fine-Grain Access Control for Securing Shared Resources in Computational Grids," *Proc. IPDPS*.
[6] G. Smith and D. Volpano, "Secure Information Flow in a Multithreaded Imperative Language," *Proc. ACM Symp. Principles Programming Languages*, Jan. 1998, pp. 355–64.

[7] D. S. Wallach, A. W. Appel, and E. W. Felten, "The Security Architecture Formerly Known as Stack Inspection: A Security Mechanism for Language-Based Systems," *ACM Trans. Software Eng. and Methodology*, vol. 9, no. 4, Oct. 2000, pp. 341–78.
[8] D. K. Mulligan and A. Burstein, "Implementing Copyright Limitations in Rights Expression Languages," *2002 ACM Wksp. Digital Rights Mgmt.*, Washington, DC, Nov. 18, 2002.
[9] P. Samuelson, "Digital Rights Management {and, or, vs.} the Law," *Commun. ACM*, vol. 46, no. 4, Apr. 2003, pp. 41–45.
[10] P. Helman and G. Liepins, "Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse," *IEEE Trans. Software Eng.*, vol. 19, no. 9, Sept., 1993, pp. 886–901.
[11] J. M. Morse *et al.*, "Verification Strategies for Establishing Reliability and Validity in Qualitative Research," *Int'l. J. Qualitative Methods*, vol. 1, no. 2, 2002.
[12] Y. Xiao, "Accountability for Wireless LANs, Ad Hoc Networks, and Wireless Mesh Networks," *IEEE Commun. Mag.*, vol. 46, no. 4, Apr. 2008, pp. 116–26.
[13] Y. Xiao, D. Takahashi, and K. Meng, "Accountability Using Flow-Net: Design, Implementation, and Performance Evaluation," to be submitted for publication.
[14] D. Takahashi and Y. Xiao, "Retrieving Knowledge from Auditing Log-Files for Computer and Network Forensics and Accountability," *Security and Commun. Net.*, vol. 1, no. 2, Feb. 2008, pp. 147–60.
[15] K. Meng, Y. Xiao, and S. V. Vrbsky, "Building a Wireless Capturing Tool for WiFi," to appear, *Security and Commun. Net.*

Biographies

YANG XIAO [SM'04] (yangxiao@ieee.org) is currently with the Department of Computer Science at the University of Alabama. He was a voting member of the IEEE 802.11 Working Group from 2001 to 2004. He serves as an Associate Editor for several journals, including *IEEE Transactions on Vehicular Technology*. His research areas are security, telemedicine, robot, sensor networks, and wireless networks. He has published more than 300 papers in major journals, refereed conference proceedings, and book chapters related to these research areas. His research has been supported by the U.S. NSF and other organizations. He is a guest professor at Jilin University (2007–2012), and an adjunct professor at Zhejiang University (2007–2009).