B-032

ネットワークアプリケーションをテストするための通信記録再生機能

A Communication Recording and Replaying Facility for Testing Network Applications

今里邦夫* 新城靖*† 鈴木真一* 板野肯三*† 加藤和彦*†
Kunio Imazato Yasushi Shinjo Shinichi Suzuki Kozo Itano Kazuhiko Kato

1. はじめに

WWW サーバやブラウザ等のネットワークアプリケーションは，セキュリティ上の脆弱性を含んでいる場合がある． 脆弱性を修正するために，システム管理者はネットワークアプリケーションの更新作業を行う． このような更新作業は，危険性を伴う． **世界 OS**[1] を用いることで， 更新前の状態に戻すことができる．しかし，ネットワークアプリケーションの挙動をテストするにあたって不十分であった．例えば，管理者はサーバをテストするために手動でクライアントを操作するしかなかった．

この論文では，安全にアプリケーションをテストするための通信記録再生機能について述べる． 通信記録再生機能とは，まず古いアプリケーションの通信を記録し，新しいアプリケーションを実行する時に，保存した通信記録から通信を再生する機能である．これにより，管理者はネットワークアプリケーションを手動で操作することなく，何度でも同じパラメタでテストし，その結果を比較できるようになる．

2. 関連研究

ReVirt[2] は，仮想計算機の入出力を記録し，再生することができる．ReVirt では，チェックポイントを設け，記録を開始する．また，記録された入出力をチェックポイントから順に再生することができる．しかしながら，そのまま再生するだけではアプリケーションのバージョンの違いに対応できない． 本研究ではプロトコルに固有の入出力を解釈することで，記録したメッセージとは一部異なる場合であっても再生することができるようにする．

3. 通信記録再生機能の提案

現状では，テストを効率的に行うための方法は提供されていない．管理者が実際にクライアントまたはサーバを立ち上げて，その動作を手動で確認する必要がある．例えば，WWW サーバをテストする場合には，管理者がブラウザを操作してサーバにアクセスしその出力を確認しなくてはならない．

確認作業を支援するため，本研究では通信記録を再生する機能を提案する (図 1)．この機能を利用するには，まず古いネットワークアプリケーションを実行し，その通信内容をファイルに記録する．更新作業後に，新しいネットワークアプリケーションを実行し，入力としてファイルの内容を与え，ネットワークアプリケーションの出力をファイルのものと比較する．

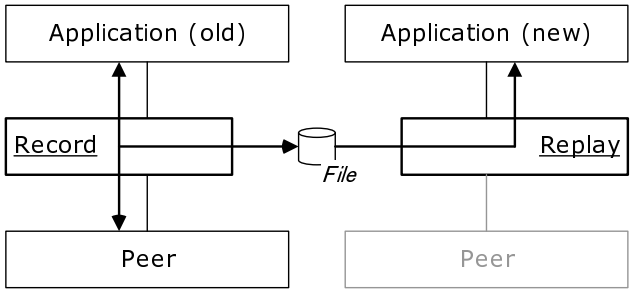

図 1 提案する通信記録再生機能

4. 通信記録再生機能の実現

通信記録再生機能を実現するにはプロキシを使う方法が考えられる．この方法の利点は，1 台のプロキシで複数のホストの通信を記録し再生できることである．しかし，この方法ではプロキシを経由しないシステムコールを記録および再生することはできない．例えば，WWW サーバのアクセス制御のテストを行いたい場合，`getpeername()` などのシステムコールを再現する必要があるが，プロキシではそのようなローカルホスト上で実行されるシステムコールを再現することができない．

そこで，本研究では，**システムコールの捕捉**により，通信の記録・再生機能を実現する．この方法の利点は，ローカルホスト上で実行されるシステムコールであっても再現できることである．

4.1 構成

本機能は図 2 で示すモジュールにより実現される．

捕捉モジュール(Systemcall-Trace Module) は，システムコールの捕捉を行う．捕捉対象とするシステムコールは，入出力 (`read`, `write` など)，ソケットの作成・削除 (`socket`, `close`)，ソケットの操作 (`bind`, `connect`, `accept`, `select`, `getpeername` など), その他 (`gettimeofday` など) である．

メインモジュール(Main Module) は，記録・再生すべきシステムコールか否かの判断や，4.2 節で述べる記録ファイルとの入出力を行う．**プロトコル固有モジュール**(Protocol Specific Module) は，プロトコルごとの処理を行う置き換え可能なモジュールである．詳しくは 4.5 節で述べる．

* 筑波大学システム情報工学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba
† 独立行政法人科学技術振興機構戦略的創造研究推進事業
Core Research for Evolutional Science and Technology, Japan Science and Technology

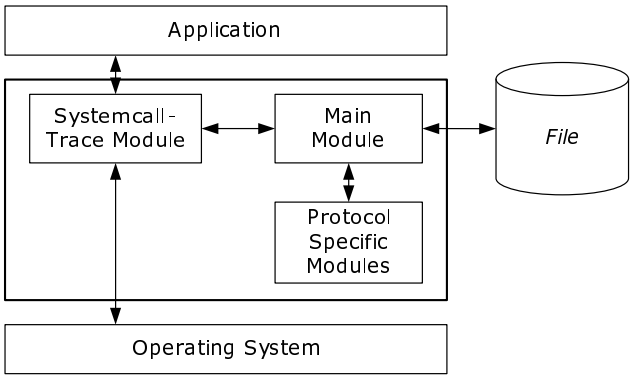

図 2　通信記録再生機能の構成

4.2　記録ファイル

記録ファイルは，レコードの集まりである．レコードには，**通信レコード**と**システムコールレコード**がある．通信レコードは，**基本ヘッダ**，**通信ヘッダ**，**プロトコルヘッダ**および**ボディ**により構成される． ボディは送受信したメッセージである．

基本ヘッダは，レコードの型，レコード長，開始時刻，終了時刻を含む． **通信ヘッダ**は，送信元アドレス情報，宛先アドレス情報，方向，コネクション ID，エラー番号，メッセージ長を含む． **プロトコルヘッダ**はプロトコル固有モジュールによって埋め込まれる可変長のヘッダである．

システムコールレコードは，基本ヘッダに加え，システムコールの引数および結果とエラー番号を含む．

4.3　通信の記録

メインモジュールはシステムコールを捕捉し，OS による処理が完了した時点で，4.2 節で示したレコードをファイルに記録する．

メインモジュールは，プロトコルモジュールと連携し，以下の処理を行う．

- 記録すべきシステムコールかどうかプロトコル固有モジュールに問い合わせる．
- 通信メッセージをプロトコル固有モジュールに渡し，レコードを作成する．
- レコードを記録ファイルに書き出す．

4.4　通信の再生

メインモジュールはシステムコールを捕捉し，OS にシステムコールの処理を行わせず，4.2 節で示したレコードの内容からシステムコールを再現する．記録ファイルからレコードを抜き出した後，以下の処理を行う．

- 再生すべきシステムコールかどうかプロトコル固有モジュールに問い合わせる．
- レコードをプロトコル固有モジュールに渡し，メッセージを再構築する．
- 再構築後のメッセージをアプリケーションの受信バッファにコピーする．
- システムコールの結果をアプリケーションに返す．

4.5　プロトコル固有モジュール

プロトコル固有モジュールは SMTP や HTTP などアプリケーション層のプロトコルに固有の処理を行う．例えば HTTP ヘッダの記録はするが，HTTP ボディの記録は行わないことを考える．記録時には，メインモジュールから受け取った通信メッセージのうち，HTTP ヘッダのみを残し，HTTP ボディを削除する．さらにプロトコル固有ヘッダとして，HTTP ヘッダサイズおよび HTTP ボディサイズ (Content-Length) を埋め込む．再生時には，まず記録されたレコードの HTTP ヘッダを再構築する．次に HTTP ボディサイズから擬似的にボディを再構築する．最後にメインモジュールに再構築したメッセージを返す．

4.6　結果の比較

4.4 節で述べた仕組みにより，ネットワークアプリケーションの挙動を再現する．再現時にネットワークアプリケーションの出力を遠隔ホストに出力する代わりに，通信レコードの出力と比較することでその動作を確認する．比較する情報はプロトコルごとに異なる．例えば HTTP の場合に，時刻等の違いを無視するようにする．また SMTP の場合に，メッセージ ID の違いやヘッダの順序の違いを無視するようにする．プロトコル固有モジュールは，プロトコルを解釈し，テストに必要な情報を差分として表示する．

5.　おわりに

ネットワークアプリケーションをテストするための通信記録再生機能について述べた． 現在，動的リンクライブラリを用いてシステムコールを捕捉し，記録および再生を行っている．この方法により，単純なネットワークアプリケーションについて通信の記録を再生することができた．今後は，プロトコル固有モジュールの作成と，より複雑なネットワークアプリケーションの記録および再生を行っていく．

参考文献

[1] 石井孝衛，新城靖，板野肯三: プロセストレース機能を用いた世界 OS の実現, 情報処理学会論文誌, Vol.43, No.6, pp.1702-1714 (2002).

[2] George W.Dunlap, Samuel T.King, Sukru Cinar, Murtaza A.Basray, and Peter M.Chen: “ ReVirt: VM Logging and Replay”, In Proceedings of the 2002 Symposium on Operating Systems Design and Implementation(OSDI), pp.211-224 (2002).