ストリーム処理システムを用いたマルウェア検知基盤システム

MWS2012

筑波大学 情報科学類 4年
北川データ工学研究室
大桶　真宏

Outline

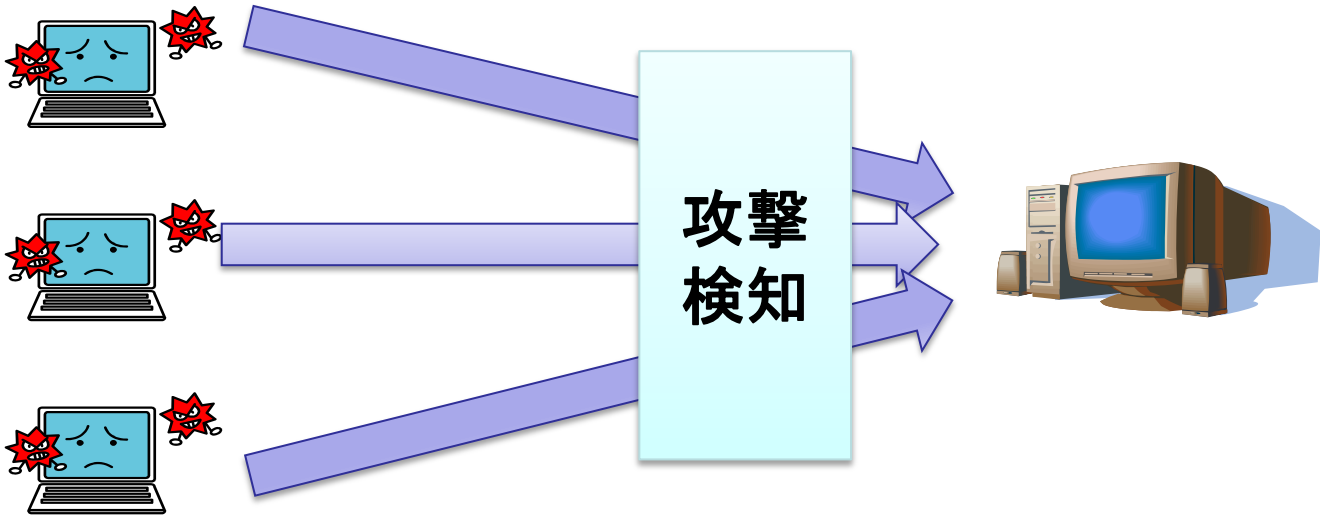
- 研究概要
- Change Point Detection(CPD)
- ストリーム処理システムSS*
- まとめと課題

Outline

- 研究概要
- Change Point Detection(CPD)
- ストリーム処理システムSS*
- まとめと課題

背景

- 多数のマルウェアが現存, 増加を続けている
- 感染したコンピュータはネットワーク上の他のコンピュータへ攻撃
- ネットワークパケットのヘッダ情報からマルウェアの攻撃を検知する手法が存在



実装と問題

- マルウェアの攻撃を検知する精度を向上させるために複数の手法を利用
- 各手法を個別にプログラムとして実装

問題点

性能:入力データを各手法に複製することが必要

開発:出力インターフェースを統一することが必要

- 各手法を単一のプログラムとして実装

問題点

開発:開発言語の柔軟性

管理:複数の手法を管理する方法が必要

提案システム

- ストリーム処理システムに攻撃検知の手法を実装
  - 検知手法を組み込み関数として実装
  - 入力ストリームデータに検知手法を適用する問合せを記述することで検知手法を実行可能
  - 複数問合せの同時実行が可能

問題点の解決

- 性能:
  - 入力ストリームデータは各問合せにプロセス内で複製
- 開発:
  - 出力インターフェースはタプルストリームに統一
  - 各検知手法は組み込み関数として実装可能
- 管理:
  - 各問合せの確認や実行/停止は容易

Outline

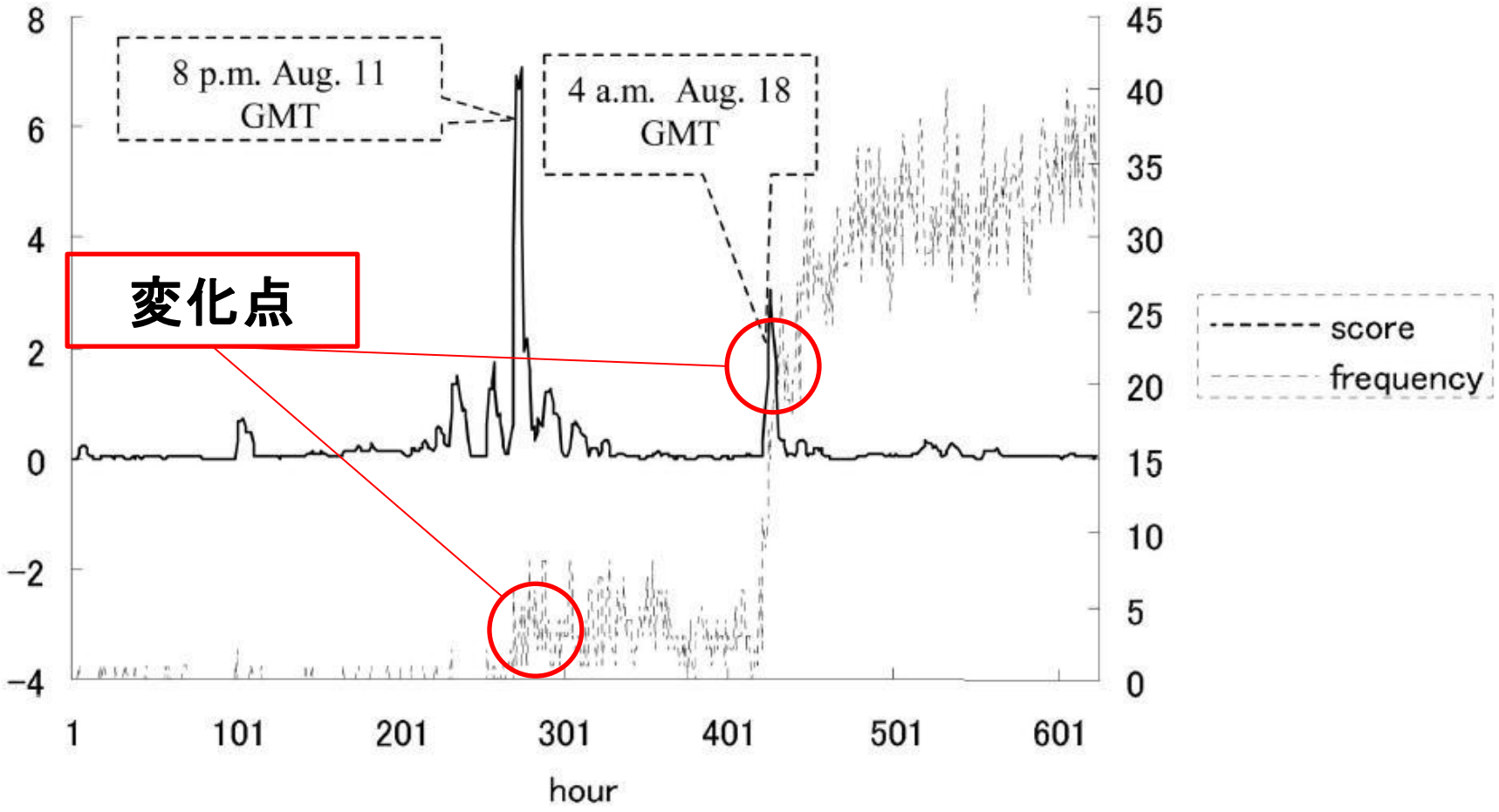
- 研究概要
- Change Point Detection(CPD)
- ストリーム処理システムSS*
- まとめと課題

CPD概要

- 時系列データから異常検知
  - ワームの発生やDoS攻撃時にはトラフィックの変化
- 2段階学習により外れ値と変化点を検出
- ARモデルを採用
  - 独自のSDARアルゴリズムで学習
    - 非定常な時系列データに対応
    - オンライン学習に対応し，計算コストの低減

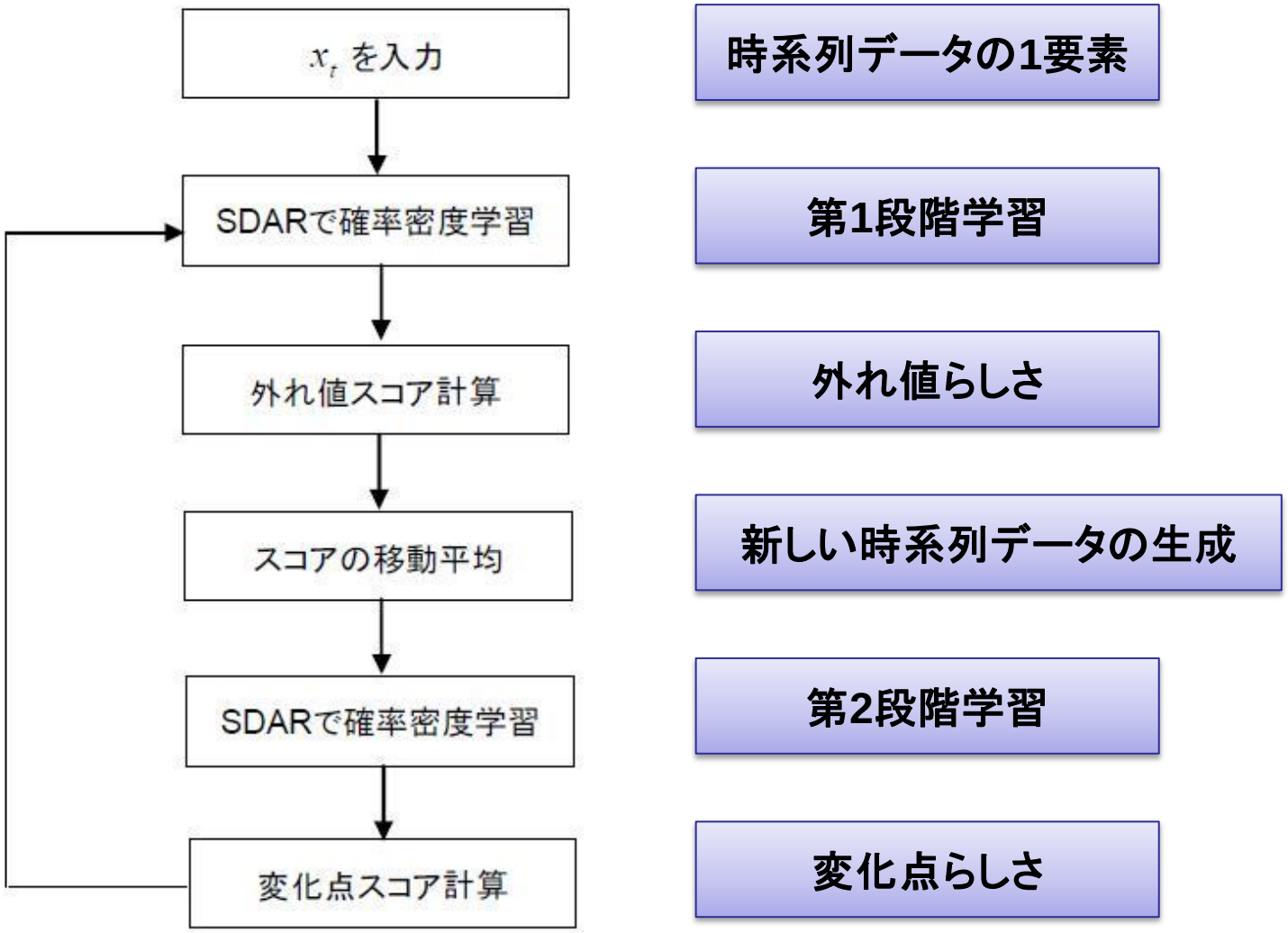
[1] J. Takeuchi and K. Yamanishi, “A Unifying Framework for Detecting Outliers and Change Points from Time Series,” IEEE transactions on Knowledge and Data Engineering, , pp.482-492, 2006.

CPD概要



[1] J. Takeuchi and K. Yamanishi, “A Unifying Framework for Detecting Outliers and Change Points from Time Series,” IEEE transactions on Knowledge and Data Engineering, , pp.482-492, 2006.

CPD概要


x_t を入力
SDARで確率密度学習
外れ値スコア計算
スコアの移動平均
SDARで確率密度学習
変化点スコア計算
時系列データの1要素
第1段階学習
外れ値らしさ
新しい時系列データの生成
第2段階学習
変化点らしさ

ARモデル

- 時系列データに対する典型的な統計モデル

k次のARモデル

$$x_t = A_k x_{t-1} + A_{k-1} x_{t-2} + A_{k-2} x_{t-3} + \ldots + \varepsilon$$

x_t :時系列データ　　t :時間

パラメータ

ホワイトノイズ
(期待値0 分散Σ)

直前までのデータからパラメータを推定 ➡ **現在のデータを予測**

SDARアルゴリズム概要

- Sequentially Discounting AR model learning
- ARモデルのパラメータを推定
- ARモデルの問題点
  - パラメータの推定にバッチ学習方式
  - 時系列データが定常という仮定
- ARモデルの学習にオンライン学習・忘却機能を追加
  - 計算コストの低減
  - 非定常なデータに対応

SDARアルゴリズム

$$I = \sum_{i=1}^{t} (1-r)^{t-i} \log p(x_i \mid x_1 x_2 \ldots x_{i-1}, A_1, A_2 \ldots, A_k, \mu, \Sigma)$$

x_i の確率密度関数

パラメータ

- *I*が最大になるようなパラメータを推定
- 忘却率rにより過去の重みが減少(0 < r < 1)
- 過去のデータの影響を小さくすることで
  非定常なデータに対応

外れ値スコア

SDARアルゴリズムで学習したモデルから確率密度を計算

$$p_{t-1}(x_t) := p(x_t \mid x_{t-k}x_{t-k+1}\ldots x_{t-1}, A_1, A_2 \ldots, A_k, \mu, \Sigma)$$

- 確率密度が高い ⇨ 予測した値と**近い**
- 確率密度が低い ⇨ 予測した値と**外れている**

$$Score(x_t) = -\log p_{t-1}(x_t)$$

外れ値スコア

スコアが大きいほど外れ値である可能性が高い

スコアの移動平均

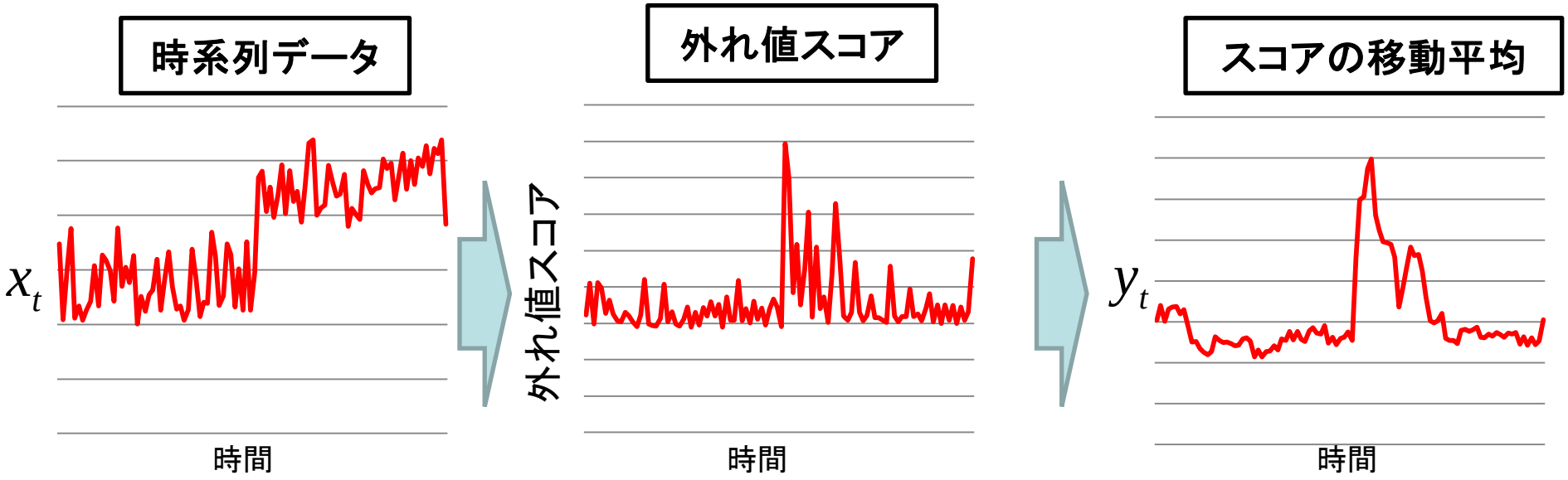
$$y_t = \frac{1}{T}\left(\sum_{i=t-T+1}^{t} Score(x_i) \right)$$

T:移動時間

新しい
時系列データ

現在時刻tからT時間前までの外れ値スコアの平均

外れ値らしさを平滑化



変化点スコア

新しい時系列データ y_t にSDARアルゴリズムを適用

第2段階学習

$$q_{t-1}(y_t) := q(y_t \mid y_{t-k'} y_{t-k'+1} \cdots y_{t-1}, A_1', A_2' \ldots, A_{k'}', \mu', \Sigma')$$

$$Score(t) = \frac{1}{T} \sum_{i=t-T+1}^{t} \left(-\log q_{i-1}(y_t)\right)$$

変化点スコア

スコアが高いほど変化点である可能性が高い

実験

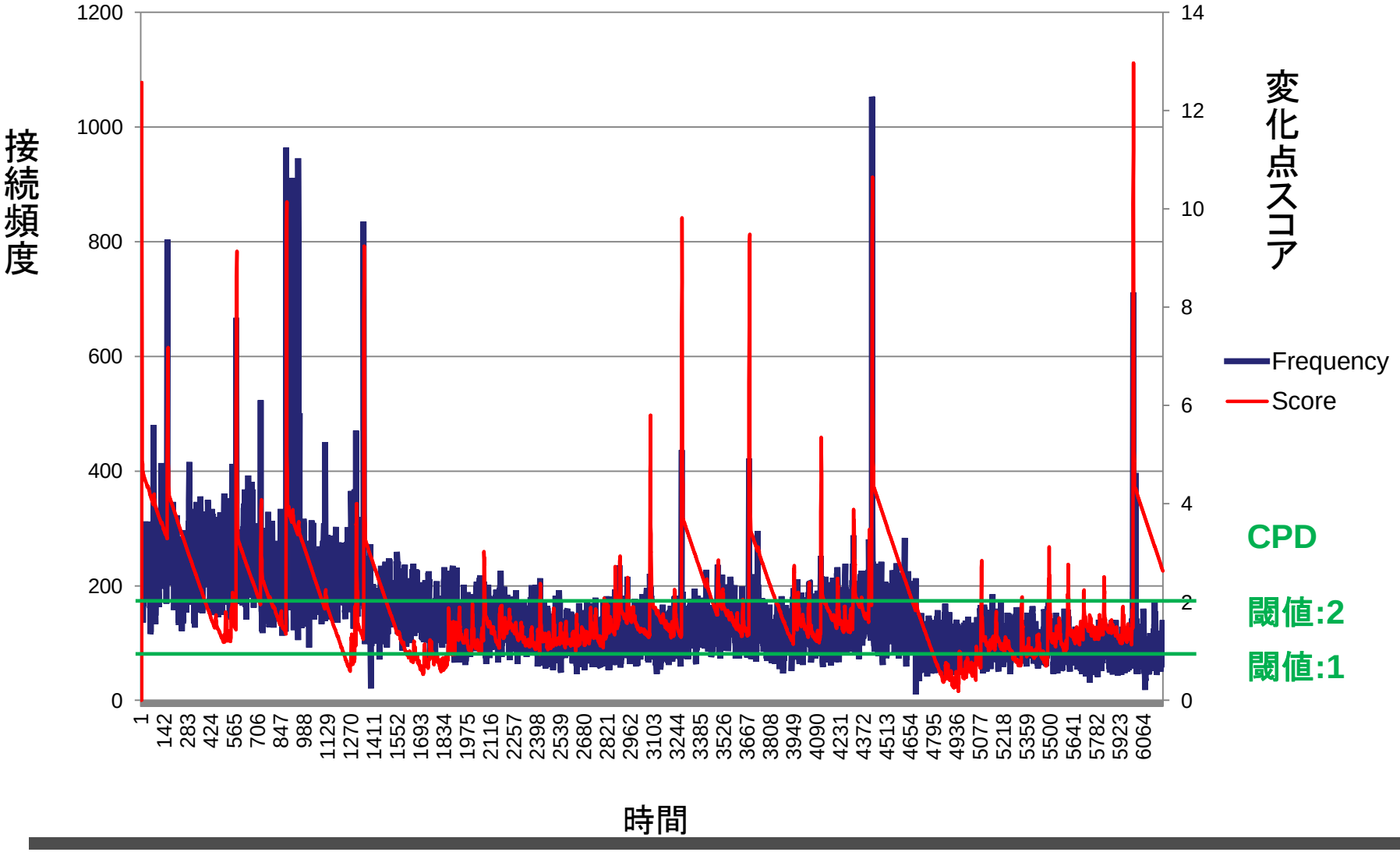
- あるサーバのPort445に対する毎時間の接続頻度の時系列にCPDを適用し, 閾値を超えた変化点を検知
- IIJ MITF DATAset 2012(MWS2012)
  - Honeypot100台への通信情報のログ(ipcomm)
  - Honeypot100台への攻撃情報(マルウェア情報)のログ(mal)
  - 2011年7月11日～2012年4月30日
- パラメータの設定
  - 第1段階:2次のARモデル,忘却率0.005
  - 第2段階:3次のARモデル,忘却率0.02
  - 移動平均:T=5

[2] MWS2012実行委員会, 研究用データセット MWS 2012 Datasetsについて, http://www.iwsec.org/mws/2012/about.html

実験:照合方法

- 変化点の時間帯のipcommデータとmalデータを利用
  - ipcommデータのsrcipとmalデータのsrcipを照合
    - YYYY-MM-DD HH:00:00に含まれるsrcip
    - srcip 1件につき1件とカウント
  - 該当するsrcipのmalデータがあるかどうか
- ipcommデータには354490件
- malデータには343356件

実験:結果


接続頻度
変化点スコア
時間
Frequency
Score
CPD
閾値:2
閾値:1
0
200
400
600
800
1000
1200
0
2
4
6
8
10
12
14
1
142
283
424
565
706
847
988
1129
1270
1411
1552
1693
1834
1975
2116
2257
2398
2539
2680
2821
2962
3103
3244
3385
3526
3667
3808
3949
4090
4231
4372
4513
4654
4795
4936
5077
5218
5359
5500
5641
5782
5923
6064

実験:結果

- 閾値2:
  - 検知した件数:130751件
  - malデータに含まれる件数:126835件
  - Recall:37%, Precision:97%
- 閾値1:
  - 検知した件数:312388件
  - malデータに含まれる件数:302498件
  - Recall:96%, Precision:96%

実験:考察

- 閾値が低ければ低いほどRecallが高い
  - 全ipcommデータと照合すると
    Recall:99%, Precision: 99%
    - データのほとんどがマルウェアによる攻撃
  - データを多く含むほうがRecallが高くなる
- マルウェア攻撃ではない定常状態の
  データを含むデータセットが欲しい

Outline

- 研究概要
- Change Point Detection(CPD)
- ストリーム処理システムSS*
- まとめと課題

ストリーム処理システム

- ストリームデータをリレーショナルスキーマで
  モデル化
- 問合せはシステムに登録され, データが到着
  する度に実行
- ストリームデータに対し, 集約演算等を適用可能
- 複数問合せを同時実行可能
- 処理はすべてメモリ上で行われる
- RDBMSに比べ, ストリームデータを高速に
  処理可能

SS*

- ストリーム処理システムSS*はStreamSpinnerの後継
- SQLライクな言語により問合せを記述
- 組み込み関数としてマルウェア検知手法を実装可能
- run/stopコマンドにより問合せを起動/停止
- 問合せの中身を確認可能

[10]StreamSpinner: http://www.streamspinner.org/

パケット分析例(1/4)

- 全パケットの状況は？
  - フルスキャン
- 各port のアクセス状況は？ ***1**
  - SYN=Yes & ポート毎のアクセス数/秒
- 各port の異常度は？ ***2**
  - SYN=Yes & ポート毎の異常度/秒

s-negi

gs-syn-negi

gs-cpd-syn-negi

***1** “Enabling Real Time Data Analysis”, Divesh Srivastava (AT&T Labs), et, al. Keynote talk, VLDB 2010.
***2** “An Incident Analysis System NICTER and Its Analysis Engines Based on Data Mining Techniques”, Daisuke Inoue (NICT), et, al. ICONIP (1) 2008: 579-586
***3 Negi:**https://github.com/westlab/negi

パケット分析例(2/4)

- パケットの状況は？
  - フルスキャン

s-negi

```
select *
from negi
```

パケット分析例(3/4)

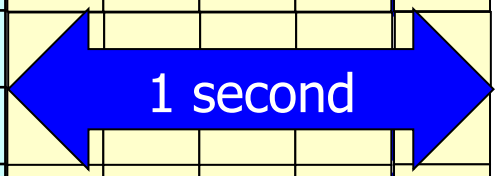
- 各port のアクセス状況は? ***1**
  - ポート毎のアクセス数/秒
  - SYN=Yes

gs-syn-negi

```
select dst_port,
       count(dst_port)
from  negi[1 sec]
group by dst_port
where syn $= true;
```

22:	2
80:	2
15:	1

src_ip					
dst_ip					
src_port					
dst_port	22	80	15	80	22
seq_no					
packet_size					
timestamp					
protocol					
ack					
fin					
syn	T	T	T	T	T
urg					
push					
reset					
content					



***1** “Enabling Real Time Data Analysis”, Divesh Srivastava (AT&T Labs), et, al. Keynote talk, VLDB 2010.

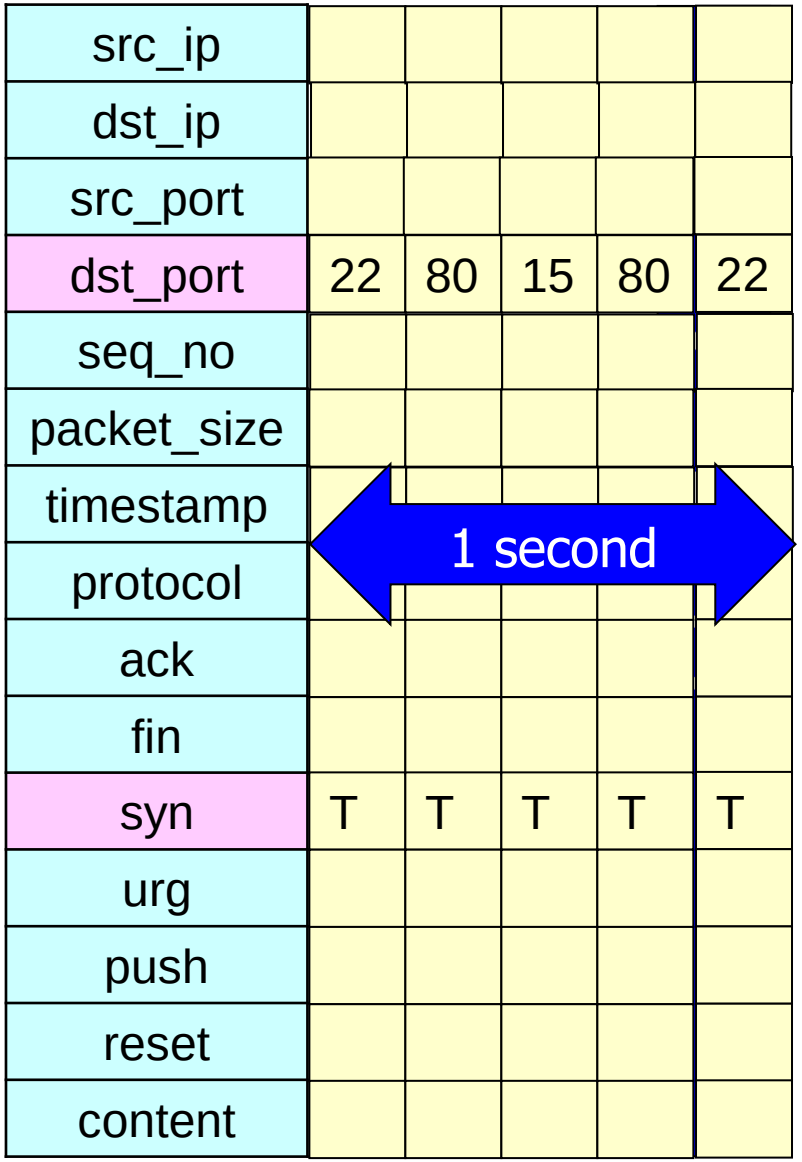
パケット分析例(4/4)

- 各portアクセスの異常度は? *2
  - ポート毎のアクセス異常度/秒
  - SYN=Yes

gs-cpd-syn-negi

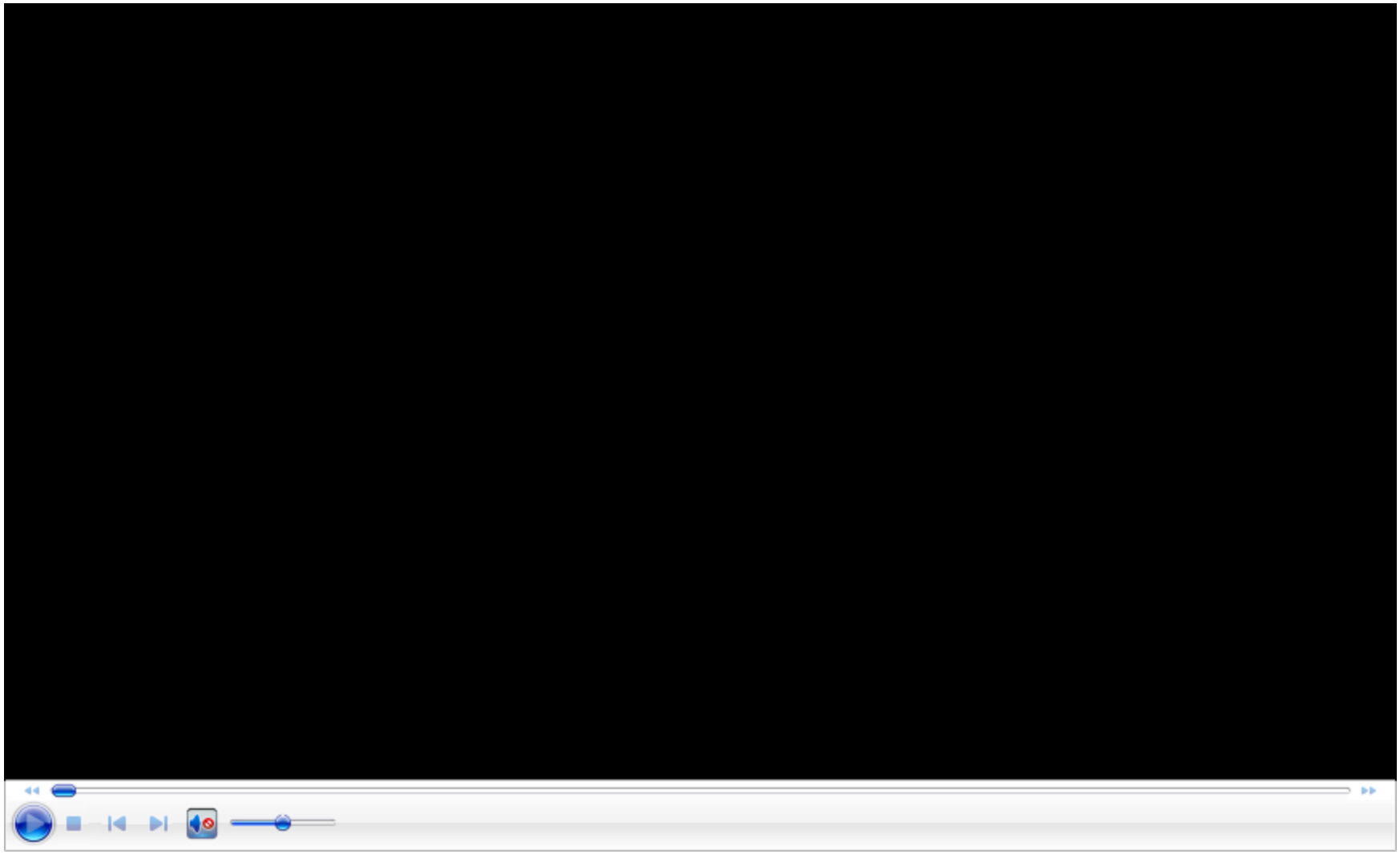
```
select dst_port,  cpd()
from  negi[1 sec]
group by dst_port
where syn $= true;
```

22:	1.33
80:	2.44
15:	1.22



*2 “An Incident Analysis System NICTER and Its Analysis Engines Based on Data Mining Techniques”, Daisuke Inoue (NICT), et, al. ICONIP (1) 2008: 579-586

デモ



*4:慶応大学 西研究室 http://www.west.sd.keio.ac.jp/

Outline

- 研究概要
- Change Point Detection(CPD)
- ストリーム処理システムSS*
- まとめと課題

まとめと課題

- まとめ
  - ストリーム処理システムを用いた
    マルウェア検知基盤システムを提案
    - 検知手法を組み込み関数で実装
    - SQLライクな言語により検知手法を実行可能

- 課題
  - CPDの同時実行における計算の共有化
  - 検知手法を増やす
  - 組み込み関数の開発言語を増やす
    - 現在はC++をサポート