プロトコルの DoS 攻撃耐性解析のための計算体系における時間モデル

Time Model of Computational System for DoS attack resistance of Protocols

西崎真也 † 池田立野 † 日高武尊 †*

Shin-ya Nishizaki Ritsuya IKEDA Takeru HIDAKA

† 東京工業大学 大学院情報理工学研究科

Department of Computer Science, Tokyo Institute of Technology

nisizaki@cs.titech.ac.jp

サービス不能攻撃 (DoS 攻撃) とは，攻撃対象のシステムがサービスを提供できないようにしたり，システムそのものをダウンさせたりする攻撃である．DoS 攻撃には，さまざまな種類のものが存在するが，そのうちの代表的なものの一つに SYN あふれ攻撃があり，これは，TCP コネクションを確立するための 3 ウェイハンドシェイクに必要なコストの差を利用したものである．本論文では，pi 計算と spi 計算を基礎とした DoS 攻撃への耐性の記述と解析のための形式体系である型付 Spice 計算に時刻の概念を取り入れた時刻つき Spice 計算を提唱する．これは DoS 攻撃にはある一定時間のうちに大量のパケットが送られてくるという特徴に着目し，計算のコストとそれが発生する時刻との関係をとらえることを可能にした計算体系である．この計算体系では，時刻は計算処理の種別に応じて階層化され時間モデルで記述されている．そして， DoS 攻撃の記述と解析の例として，実際に 3 ウェイハンドシェイク，SYN あふれ攻撃が行われている場合を時刻付き Spice 計算を用いて記述・解析した．

1 はじめに

まず最初に，研究の背景となる諸概念について紹介する．

サービス不能攻撃

サービス不能攻撃 (Denial-of-Service attack, DoS-attack) は，ソフトウェアにおける安全性をおびやかす典型的なものであり，正規のユーザーが計算機やネットワークの使用を不可能においこむような攻撃である．典型的な DoS 攻撃の例としては，TCP における SYN あふれ攻撃 (SYN flooding attack)[SKK+97] があげられる．

Spice 計算

プロトコルのサービス拒否攻撃に対する耐性について形式的な議論したものとしては，Meadows の研究 [Mea99] がある。これは，Alice-Bob 形式のプロトコル記述に実際に行われる計算の処理を付記することによりプロトコルでの計算のコストを明示化し，DoS 攻撃に関する耐性を解析しようとしたものである。

一方，ネットワークプロトコルを記述する体系としては，さまざまなものが提唱されてきた。それらのなかから，spi 計算 [AG97a, AG97b] を基にして，spice 計算 (secure pi-calculus with cost-estimation)[TNI04, 冨池西] を提唱した．spice 計算は，pi 計算，spi 計算を拡張したもので，プロセスの計算における計算コストが，サーバーやクライアントの計算機において，どのように費されるかを明示的に表現できるように，システムにおける計算機構成を型として形式化したものである．書き換えスタイルの操作的意味論があたえられており，プロセスに対する型付けの情報を利用することにより，計算の進行における計算コストを区別するようになっている．記憶コストは，サービス不能攻撃耐性を測る場合，各種の計算コストのうちで最も重要となるのだが，spice 計算では，記憶領域の開放を明示的におこなうようにした．

本研究の目的

DoS 攻撃においては，攻撃対象者においてある特定の時刻において，その前後の一定時間内に負荷が増大することが深刻になる．本研究では，Spice 計算

*本研究は東京工業大学でなさられた．現在は岡田眼科勤務

の操作的意味論である遷移関係に時刻の概念を導入し，コストと時刻の関係を明示的にすることを目的とする．

2 Spice 計算と時間モデル

2.1 Spice 計算の構文

まず，値と項の構文を以下の文法により定義する．項はデータを表すもので，項を評価した計算結果が値である．

$$V, U, W ::= n \mid i \mid x \mid (V_1, \ldots, V_n) \mid hash_V$$

V, U, W は値を表す，n，i，x は，それぞれ名前，整数，変数を表す．

$$\begin{array}{rcl} M, N & ::= & V \mid (M + N) \mid [M_1, \ldots, M_n] \\ & \mid & \mathsf{hash}(M) \end{array}$$

M, N は項を表す．$(M_1, \ldots, M_n)$ は n 個からなる列を表し，これが評価された値の列が，$[M_1, \ldots, M_n]$ はである．$\mathsf{hash}(M)$ は，項 M の値のハッシュ値を表す項であり，(M の評価値が V になるとすると) この項の評価結果であるハッシュ値は $hash_V$ となる．

次に，プロセスを定義する．

$$\begin{array}{rcl} P, Q, R & ::= & \mathsf{out}\ M\ \langle N \rangle; P \mid \mathsf{inp}\ M\ (x); P \\ & \mid & (P \mid Q) \mid \mathsf{new}(n); P \mid \mathsf{repeat}\ P \\ & \mid & \mathsf{stop} \mid \mathsf{store}\ x = M; P \mid \mathsf{free}\ x; P \\ & \mid & \mathsf{split}\ [x_1, \ldots, x_n]\ \mathsf{is}\ M\ \mathsf{err}\{R\}; P \\ & \mid & \mathsf{match}\ M\ \mathsf{is}\ N\ \mathsf{err}\{P\}; Q \end{array}$$

$\mathsf{out}\ M\ \langle N \rangle P$ は，ポート M を経由してメッセージ N を送信し，プロセス P が続くというプロセスである．$\mathsf{inp}\ M\ (x); P$ は，ポート M を経由してメッセージを受け取り，それを変数 x に束縛し，プロセス P を続けて実行するというプロセスである．$(P \mid Q)$ は，P と Q とのパラレルに実行するプロセスである．$\mathsf{new}(n); P$ は，プロセス P における名前 n の制限を意味する．stop は終了状態のプロセスである．次にエージェントと呼ばれる構文を定義する．

$$A, B ::= P \mid (x)P \mid (\nu n_1 \cdots n_j)\langle M \rangle P$$

$(x)P$ と $(\nu n_1 \cdots n_j)\langle M \rangle P$ は，各々，受信プロセス，送信プロセスの遷移における中間状態を表す．

2.2 Spice 計算の時間モデル

本研究における時間モデルは，プロセスの実行経過において，各時刻にどのようなコストが発生するのかということをとらえることを目的としている．したがって，時間の構造は線形なものとする．ある微小時間を単位時間とし，時刻の経過をその微小な単位時間の累積としてとらえるというのが一般的である．本研究では，単位時間の粒度に，計算処理の種類に応じて，いくつかの階層を設ける．

本論文では，以下のような時間階層は 3 階層を設けることとする．(1) 最も細かい時間単位は，四則演算やハッシュ値の計算のようなプロセス内計算の実行時間に対応する．(2) 中粒度の時間単位は，正常な通信の所要時間に対応する．(3) 粗粒度の時間単位は，通信失敗のタイムアウトの所要時間に対応する．時刻は，自然数の三つ組

$$(t_1, t_2, t_3)$$

とし，辞書式順序により順序付けする．すなわち，t_1 が粗粒度の時刻である．例えば，$(3, 2, 4) > (2, 5, 10) > (2, 5, 9)$ である．また，時刻と時間，時間と時間との間の加減などの演算は，3 次元数ベクトルのものとする．

2.3 Spice 計算の型付け

プロセス P が実行される計算機環境を表現する計算機型と呼ばれるもので，以下の文法により定義される．

$$\mathcal{A}, \mathcal{B} ::= \mathbf{a}::\{x_1, \ldots, x_n\} \mid (\mathcal{A} \mid \mathcal{B})$$

例えば，$\mathbf{a}::\{x, y\} \mid \mathbf{b}::\{z, w\}$ は，プロセスが計算機 $\mathbf{a}$ と計算機 $\mathbf{b}$ 上で実行され，各計算機では，変数 x, y と変数 z, w がメモリアロケートされていることを表現している．

Spice 計算における型付けは，変数におけるメモリアロケーションの整合性の保証と，計算において発生するコストがどの計算機に由来するのかという情報を与える役割をはたしている．

$$\frac{\mathbf{a}::\mathcal{V} \rhd P \quad \mathcal{V} \supseteq fv(M) \quad \mathcal{V} \supseteq fv(N)}{\mathbf{a}::\mathcal{V} \rhd \mathsf{out}\ M\ \langle N \rangle; P}$$

$\mathcal{V}$ は変数の集合を表し，$fv(M)$ は，項 M に出現する変数全体の集合を表す．

$$\frac{\mathbf{a}{::}(\{x\}\cup\mathcal{V})\rhd P \quad \mathcal{V}\not\ni x \quad \mathcal{V}\supseteq fv(M)}{\mathbf{a}{::}\mathcal{V}\rhd \mathsf{inp}\ M\ (x);P}$$

入力プロセスの $\mathsf{inp}\ M\ (x);P$ の型とその部分プロセス P の型とを比較すると，変数 x が追加されている点が異なるが，これは，プロセス P の実行が開始される時点で，変数 x が割当てられていることを意味している．

$$\frac{\mathcal{A}\rhd P \quad \mathcal{B}\rhd Q}{(\mathcal{A}\mid\mathcal{B})\rhd(P\mid Q)}$$

$$\frac{\mathcal{A}\rhd P}{\mathcal{A}\rhd \mathsf{new}(n);P} \qquad \frac{\mathbf{a}{::}\emptyset\rhd P}{\mathbf{a}{::}\emptyset\rhd \mathsf{repeat}\,P}$$

この規則は，プロセスの複製は変数が割当てられていない状態の計算機上でのみ許されることを意味している．この制限により，プロセスの複製が実行されるときには，変数の格納領域は空になっていて，このとき，変数領域の複製は引き起されない．このように定義した理由は，プロセス複製のコスト付けを単純化するためである．

$$\frac{}{\mathbf{a}{::}\emptyset\rhd \mathsf{stop}}$$

この規則は，プロセス正常終了時には，変数が割当てられていない状態でなければならないことを意味している．すなわち，この規則は，正常終了するときには，割当てられた記憶領域はすべて，free で開放されることを強制している．

$$\frac{\mathbf{a}{::}(\{x\}\cup\mathcal{V})\rhd P \quad \mathcal{V}\not\ni x \quad \mathcal{V}\supseteq fv(M)}{\mathbf{a}{::}\mathcal{V}\rhd \mathsf{store}\ x=M;P}$$

$$\frac{\mathbf{a}{::}\mathcal{V}\rhd P \quad \mathcal{V}\not\ni x}{\mathbf{a}{::}(\{x\}\cup\mathcal{V})\rhd \mathsf{free}\ x;P}$$

$$\frac{\mathbf{a}{::}\mathcal{V}\rhd P \quad \mathbf{a}{::}\mathcal{V}\rhd Q \quad \mathcal{V}\supseteq fv(M) \quad \mathcal{V}\supseteq fv(N)}{\mathbf{a}{::}\mathcal{V}\rhd \mathsf{match}\ M\ \mathsf{is}\ N\ \mathsf{err}\{P\};Q}$$

$$\frac{\left\{\begin{array}{l}\mathbf{a}{::}(\{x_1,\ldots,x_n\}\cup\mathcal{V})\rhd P\\ \mathbf{a}{::}\mathcal{V}\rhd R\\ \{x_1,\ldots,x_n\}\cap\mathcal{V}=\emptyset\\ \mathcal{V}\supseteq fv(M)\end{array}\right.}{\mathbf{a}{::}\mathcal{V}\rhd \mathsf{split}\ (x_1,\ldots,x_n)\ \mathsf{is}\ M\ \mathsf{err}\{R\};P}$$

$$\frac{\mathcal{A}\rhd P}{\mathcal{A}\rhd(x)P} \qquad \frac{\mathcal{A}\rhd P \quad fv(M)=\emptyset}{\mathcal{A}\rhd(\nu n_1\cdots n_j)\langle M\rangle P}$$

式 M には自由変数が含まれていないことが条件になっている．後述の操作的意味論からわかるように，この具体化エージェントが発生するときには，実際には，式 M は，自由変数を含まないような値となる．

2.4 Spice 計算の時刻付きコスト意味論

pi 計算や spi 計算では，ラベル付き遷移関係により形式化された操作的意味論が与えられている．ラベルは，プロセスの実行において発生する送信や受信に関する “シグナル” である． Spice 計算では，この操作的意味論を拡張し，プロセスの実行において発生するコストを追跡することが可能とした．

pi,spi 計算 $\mathsf{inp}\ n\ (x);P \xrightarrow{n} (x)P$

Spice 計算 $\mathbf{a}{::}\mathcal{V}\vdash \mathsf{inp}\,n\,(x);P \xrightarrow{n} (x)P : \{\mathbf{a}\cdot store_x\}$

Spice 計算の遷移関係の上記の例において，“$\{\mathbf{a}\cdot store_x\}$” がこの遷移で発生するコストであり，計算機 $\mathbf{a}$ においてコスト $store$ が発生することを意味している．

コストの種類を表すコスト基底 $\mathbf{u}_1,\ldots,\mathbf{u}_j$ の整数係数の線形結合 $n_1\mathbf{u}_1+\cdots+n_j\mathbf{u}_j$ をコスト値と呼ぶ．このコスト値と計算機名とのペアの有限集合をコスト割り当てと呼ぶ．例えば，

$$\{\mathbf{a}\cdot(2\mathbf{store}+\mathbf{hash}),\mathbf{b}\cdot(\mathbf{store}+3\mathbf{match})\}$$

というコスト割り当ては，計算機 $\mathbf{a}$ において，コスト $2\mathbf{store}+\mathbf{hash}$ が発生し，計算機 $\mathbf{b}$ において，コスト $\mathbf{store}+3\mathbf{match}$ が発生していることを表している．そして，本論文では，さらにコストが発生した時刻の情報が加わる．具体的には

$$\{\mathbf{a}\cdot(2\mathbf{store}^{(0,0,2)}+\mathbf{hash}^{(0,1,1)}),\ \mathbf{b}\cdot(\mathbf{store}^{(0,1,0)}+3\mathbf{match}^{(0,0,2)})\}$$

というふうに，コスト基底にコストの発生時刻が付記される．

本論文では，操作的意味論を与える遷移関係を拡張する．遷移の前後で時刻を付記し，遷移においてなされる計算の種類に応じて時刻を進めるようにした．

$$(t_1,t_2,t_3),\mathbf{a}{::}\mathcal{V}\vdash \mathsf{inp}\,n\,(x);P \xrightarrow{n} (x)P : \{\mathbf{a}\cdot store_x^{(t_1,t_2,t_3)}\}$$

項の評価に関する遷移関係である簡約関係は

$$(t_1,t_2,t_3) \vdash P > Q : c, (t_1',t_2',t_3')$$

と表され，次の規則により定義される．(t_1,t_2,t_3) は $\vec{t}$ と略記し，(t_1',t_2',t_3') は $\vec{t'}$ と略記する．

$$\vec{t} \vdash V > V : 0, \vec{t}$$

$$\frac{\vec{t} \vdash M_1 \downarrow i_1 : c_1, \vec{t'} \quad \vec{t'} \vdash M_2 \downarrow i_2 : c_2, \vec{t''} \quad j = i_1 + i_2}{\vec{t} \vdash (M_1 + M_2) \downarrow j : c_1 + c_2, \vec{t''}}$$

$$\frac{\vec{t} \vdash M \downarrow V : c, \vec{t'}}{\vec{t} \vdash \mathsf{hash}(M) \downarrow hash_V : c + hash^{(t_1',t_2',t_3'+1)}},$$

プロセス内計算を表すプロセスの簡約関係 $\vec{t} \vdash P > Q : c, \vec{t'}$ は以下のように定義される．

$$\vec{t} \vdash \mathsf{repeat}\, P > P \mid (\mathsf{repeat}\, P) : repeat^{(t_1,t_2,t_3+1)}, (t_1,t_2,t_3+1)$$

$$\vec{t} \vdash \mathsf{inp}\ n\ (x); P\ \mathsf{err}\{Q\} > Q : \{\ \}, (t_1+1,t_2,t_3)$$

この規則は，受信プロセスが受信に失敗し，タイムアウトする処理を表現している．時刻は，最も粗い単位時間が進み，(t_1,t_2,t_3) から (t_1+1,t_2,t_3) となる．

$$\frac{\vec{t} \vdash M \downarrow V : c, \vec{t'} \quad fv(V) = \emptyset}{\vec{t} \vdash \mathsf{store}\ x = M; P > P[V/x] : c + store_x^{(t_1',t_2',t_3'+1)}, (t_1',t_2',t_3'+1)}$$

$$\vec{t} \vdash \mathsf{free}\ x; P > P : -store_x^{(t_1,t_2,t_3+1)}, (t_1,t_2,t_3+1)$$

この他に，項の照合 match に関する規則と対分解 split に関する規則がある．

$$\frac{\vec{t} \vdash M \downarrow V : c, \vec{t'} \quad \vec{t'} \vdash N \downarrow V : d, \vec{t''}}{\vec{t} \vdash \mathsf{match}\ M\ \mathsf{is}\ N\ \mathsf{err}\{P\}; Q > Q : c + d + match^{(t_1'',t_2'',t_3''+1)}, (t_1'',t_2'',t_3''+1)}$$

$$\frac{\vec{t} \vdash M \downarrow V : c, \vec{t'} \quad \vec{t'} \vdash N \downarrow W : d, \vec{t''} \quad V \not\equiv W}{\vec{t} \vdash \mathsf{match}\ M\ \mathsf{is}\ N\ \mathsf{err}\{P\}; Q > P : c + d + match^{(t_1'',t_2'',t_3''+1)}, (t_1'',t_2'',t_3''+1)}$$

$$\frac{\vec{t} \vdash M \downarrow (V_1,\ldots,V_n) : c, \vec{t'} \quad fv(V_i) = \emptyset\ (i = 1,\ldots,n)}{\vec{t} \vdash \mathsf{split}\ [x_1,\ldots,x_n]\ \mathsf{is}\ M\ \mathsf{err}\{R\}; P > P[V_1/x_1,\ldots,V_n/x_n] : c + n \times store_x^{(t_1',t_2',t_3'+1)}, (t_1',t_2',t_3'+1)}$$

$$\frac{\vec{t} \vdash M \downarrow V : c, \vec{t'} \quad V \text{ is not a pair.}}{\vec{t} \vdash \mathsf{split}\ (x_1,\ldots,x_n)\ \mathsf{is}\ M\ \mathsf{err}\{R\}; P > R : c, \vec{t'}}$$

簡約関係を複数ステップに拡張したものとして複簡約関係 $(t_1,t_2,t_3), \mathcal{A} \vdash P \gg Q : \sigma, (t_1',t_2',t_3')$ を与える．

$$\frac{(t_1,t_2,t_3), \mathcal{A} \vdash P \gg P : \{\ \}, (t_1,t_2,t_3) \quad (t_1,t_2,t_3) \vdash P > P' : c, (t_1',t_2',t_3')}{(t_1,t_2,t_3), \mathbf{a}::\mathcal{V} \vdash P \gg P' : \{\mathbf{a} \cdot c^{(t_1',t_2',t_3')}\}, (t_1',t_2',t_3')}$$

$$\frac{\begin{cases} (t_1,t_2,t_3), \mathcal{A} \vdash P \gg P' : \sigma_1, (t_1',t_2',t_3') \\ \mathcal{A} \simeq \mathcal{B} \\ (t_1',t_2',t_3'), \mathcal{B} \vdash P' \gg P'' : \sigma_2, (t_1'',t_2'',t_3'') \end{cases}}{(t_1,t_2,t_3), \mathcal{A} \vdash P \gg P'' : \sigma_1 + \sigma_2, (t_1'',t_2'',t_3''+1)}$$

$$\frac{\begin{array}{c}(t_1,t_2,t_3), \mathcal{A} \vdash P \gg P' : \sigma_1, (t_1',t_2',t_3') \\ (t_1,t_2,t_3), \mathcal{B} \vdash Q \gg Q' : \sigma_2, (t_1'',t_2'',t_3'')\end{array}}{\begin{array}{l}(t_1,t_2,t_3), (\mathcal{A} \mid \mathcal{B}) \vdash (P \mid Q) \gg (P' \mid Q') \\ \quad : \sigma_1 + \sigma_2, (t_1',t_2',t_3'+1) \sqcup (t_1'',t_2'',t_3''+1)\end{array}}$$

$$\frac{(t_1,t_2,t_3), \mathcal{A} \vdash P \gg P' : \sigma, (t_1',t_2',t_3')}{(t_1,t_2,t_3)\mathcal{A} \vdash \mathsf{new}(n); P \gg \mathsf{new}(n); P' : \sigma, (t_1',t_2',t_3')}$$

上記の型の間の二項関係 $\mathcal{A} \simeq \mathcal{B}$ は以下のように帰納的に定義される．

$$\overline{\mathbf{a}::V \simeq \mathbf{a}::W} \quad \overline{(\mathbf{a}::V) \mid \mathbf{a}::W \simeq \mathbf{a}::X}$$

$$\overline{\mathcal{A} \simeq \mathcal{A}} \quad \frac{\mathcal{A} \simeq \mathcal{A}'}{(\mathcal{A} \mid \mathcal{B}) \simeq (\mathcal{A}' \mid \mathcal{B})} \quad \frac{\mathcal{B} \simeq \mathcal{B}'}{(\mathcal{A} \mid \mathcal{B}) \simeq (\mathcal{A} \mid \mathcal{B}')}$$

プロセス間計算を表現する遷移関係として，コミットメント関係 $\vec{t}, \mathcal{A} \vdash P \xrightarrow{\alpha} A : \sigma, \vec{t'}$ がある．ここで，$\mathcal{A}$ は，計算機型である．

$$\vec{t}, \mathbf{a}::\mathcal{V} \vdash \mathsf{inp}\, n\, (x); P \xrightarrow{n} (x)P : \{\mathbf{a} \cdot store_x^{(t_1,t_2+1,t_3)}\}, (t_1,t_2+1,t_3)$$

計算機 $\mathbf{a}$ 上において，時刻 $\vec{t} = (t_1,t_2,t_3)$ にメッセージを受信すると，計算機 $\mathbf{a}$ 上でコスト $store_x^{(t_1,t_2+1,t_3)}$ が発生し，時刻は (t_1,t_2+1,t_3) に遷移する．

$$\frac{\vec{t} \vdash N \downarrow V : c, \vec{t'} \quad fv(V) = \emptyset}{\vec{t}, \mathbf{a}::\mathcal{V} \vdash \mathsf{out}\ n\ \langle N \rangle; P \xrightarrow{\overline{n}} (\nu)\langle V \rangle; P : \{\mathbf{a} \cdot c^{\vec{t'}}\}, \vec{t'}}$$

$$\frac{\vec{t}, \mathcal{A} \vdash P \xrightarrow{n} F : \sigma_1, \vec{t'} \quad \vec{t}, \mathcal{B} \vdash Q \xrightarrow{\overline{n}} C : \sigma_2, \vec{t''}}{\vec{t}, (\mathcal{A} \mid \mathcal{B}) \vdash (P \mid Q) \xrightarrow{\tau} F \mathbin{@} C : \sigma_1 + \sigma_2, \vec{t'} \sqcup (t_1'', t_2''+1, t_3'')}$$

時刻 (t_1,t_2,t_3) に計算機 $\mathcal{A}$ の下でプロセス P を 1 ステップ実行すると，シグナル n が発生し，プロセス F に遷移したときのコストが σ_1 と仮定し，また，計算

機 $\mathcal{B}$ の下でプロセス Q を 1 ステップ実行すると，シグナル $\overline{n}$ が発生し，プロセス C に遷移したときのコストが σ_2 であると仮定する．このとき，並列計算機 $\mathcal{A} \mid \mathcal{B}$ の下で，時刻 (t_1, t_2, t_3) に，並列プロセス $P \mid Q$ を 1 ステップ実行すると，シグナル τ が発生し，$F@C$ プロセスへと遷移する．このときのコストは $\sigma_1 + \sigma_2$ であり，時刻は (t'_1, t'_2, t'_3) と $(t''_1, t''_2{+}1, t''_3)$ のうち，より進んだものとなる．また相互作用 $F \mathbin{@} C$ については次の定義で説明する．

$$\frac{\vec{t}, \mathcal{B} \vdash Q \xrightarrow{\overline{n}} C : \sigma_2, \vec{t'} \quad \vec{t}, \mathcal{A} \vdash P \xrightarrow{n} F : \sigma_1, \vec{t''}(}{\vec{t}, (\mathcal{B} \mid \mathcal{A}) \vdash (P \mid Q) \xrightarrow{\tau} \ C \mathbin{@} F : \sigma_2 + \sigma_1, (t'_1, t'_2{+}1, t'_3) \sqcup \vec{t''}}$$

$$\frac{\vec{t}, \mathcal{A} \vdash P \xrightarrow{\alpha} A : \sigma, \vec{t'}}{\vec{t}, (\mathcal{A} \mid \mathcal{B}) \vdash (P \mid Q) \xrightarrow{\alpha} (A \mid Q) : \sigma, \vec{t'}}$$

$\mathcal{A}$ で表される計算機構成の下で時刻 (t_1, t_2, t_3) にプロセス P を 1 ステップ実行すると，シグナル α が発生し，プロセス A に遷移し、その処理で要するコストが σ であり，遷移した時刻が (t'_1, t'_2, t'_3) であると仮定する．このとき，並列計算機 $\mathcal{A} \mid \mathcal{B}$ の下で並列プロセス $P \mid Q$ を 1 ステップ実行すると，シグナル α が発生し，並列プロセス $A \mid Q$ に遷移する．この処理で要するコストは σ であり，時刻は (t'_1, t'_2, t'_3) に遷移する．

$$\frac{\vec{t}, \mathcal{B} \vdash Q \xrightarrow{\alpha} A : \sigma, \vec{t'}}{\vec{t}, (\mathcal{A} \mid \mathcal{B}) \vdash (P \mid Q) \xrightarrow{\alpha} (P \mid A) : \sigma, \vec{t'}}$$

$\mathcal{B}$ で表される計算機構成の下で時刻 (t_1, t_2, t_3) にプロセス Q を 1 ステップ実行すると，シグナル α が発生し，プロセス A に遷移し、その処理で要するコストが σ であり，遷移した時刻が (t'_1, t'_2, t'_3) であると仮定する．このとき，並列計算機 $\mathcal{A} \mid \mathcal{B}$ の下で並列プロセス $P \mid Q$ を 1 ステップ実行すると，シグナル α が発生し，並列プロセス $P \mid A$ に遷移する．この処理で要するコストは σ であり，時刻は (t'_1, t'_2, t'_3) に遷移する．

$$\frac{(t_1, t_2, t_3), \mathcal{A} \vdash P \xrightarrow{\alpha} A : \sigma, (t'_1, t'_2, t'_3) \quad \alpha \notin \{m, \overline{m}\}}{(t_1, t_2, t_3), \mathcal{A} \vdash \mathsf{new}(m); P \xrightarrow{\alpha} \mathsf{new}(m); A : \sigma, (t'_1, t'_2, t'_3)}$$

$\mathcal{A}$ で表される計算機構成の下で時刻 (t_1, t_2, t_3) にプロセス P を 1 ステップ実行すると，シグナル α が発生し，プロセス A に遷移し、その処理で要するコストが σ であり，遷移した時刻が (t'_1, t'_2, t'_3) であると仮定する．またこのとき，プロセス $\alpha \notin \{m, \overline{m}\}$ と仮定する．このとき $\mathcal{A}$ で表される計算機構成の下で時刻 (t_1, t_2, t_3) に名前制限プロセス P を 1 ステップ実行すると，シグナル α が発生し，プロセス A に遷移し、その処理で要するコストが σ であり，遷移した時刻が (t'_1, t'_2, t'_3) である．

エージェント F,C を各々$(x)P$, $(\nu n_1, \ldots, n_k)\langle M \rangle Q$ とする．相互作用 $F \mathbin{@} C$ とは以下のように定義されるプロセスである．相互作用 $C \mathbin{@} F$ についても，左右対照に定義される．

$$\begin{aligned}
&(x)P \mathbin{@} (\nu n_1 \cdots n_k)\langle M \rangle Q \\
&\quad \equiv \mathsf{new}(n_1) \cdots \mathsf{new}(n_k)(P[M/x] \mid Q) \\
&((\nu n_1 \cdots n_k)\langle M \rangle Q) \mathbin{@} (x)P \\
&\quad \equiv \mathsf{new}(n_1); \cdots \mathsf{new}(n_k); (Q \mid P[M/x])
\end{aligned}$$

各計算ステップの開始時刻がある時間間隔ずれると，そのステップの終了時刻もその時間間隔だけずれるという性質がある．

任意の時刻 $t_1, t_2, t_3, t'_1, t'_2, t'_3$，任意の時間 $\Delta t_1, \Delta t_2, \Delta t_3$，項 M，値 V，コスト値 c に対して，

$$\begin{aligned}
&(t_1, t_2, t_3) \vdash M \downarrow V : c, (t'_1, t'_2, t'_3) \\
\Leftrightarrow\ &(t_1 + \Delta t_1, t_2 + \Delta t_2, t_3 + \Delta t_3) \\
&\quad \vdash M \downarrow V \\
&\quad : c, (t'_1 + \Delta t_1, t'_2 + \Delta t_2, t'_3 + \Delta t_3)
\end{aligned}$$

任意の時刻 $t_1, t_2, t_3, t'_1, t'_2, t'_3$，任意の時間 $\Delta t_1, \Delta t_2, \Delta t_3$，プロセス P,Q，コスト値 c に対して，

$$\begin{aligned}
&(t_1, t_2, t_3) \vdash P > Q : c, (t'_1, t'_2, t'_3) \\
\Leftrightarrow\ &(t_1 + \Delta t_1, t_2 + \Delta t_2, t_3 + \Delta t_3) \\
&\quad \vdash P > Q \\
&\quad : c, (t'_1 + \Delta t_1, t'_2 + \Delta t_2, t'_3 + \Delta t_3)
\end{aligned}$$

任意の時刻 $t_1, t_2, t_3, t'_1, t'_2, t'_3$，計算機 $\mathcal{A}$，任意の時間 $\Delta t_1, \Delta t_2, \Delta t_3$，プロセス P,A，コスト値 σ に対して，

$$\begin{aligned}
&(t_1, t_2, t_3), \mathcal{A} \vdash P \xrightarrow{\alpha} A : \sigma, (t'_1, t'_2, t'_3) \\
\Leftrightarrow\ &(t_1 + \Delta t_1, t_2 + \Delta t_2, t_3 + \Delta t_3), \mathcal{A} \\
&\quad \vdash P \xrightarrow{\alpha} A \\
&\quad : \sigma, (t'_1 + \Delta t_1, t'_2 + \Delta t_2, t'_3 + \Delta t_3)
\end{aligned}$$

3 記述例

この章では，時刻付き Spice 計算の例を紹介する．

以下は，TCP の 3 ウェイハンドシェイクをアリスボブ記法をもちいて定式化したものである．

$$\begin{aligned} A \to B &: A, B, S_A \\ B \to A &: B, A, S_B, S_A + 1 \\ A \to B &: A, B, S_A + 1, S_B + 1. \end{aligned}$$

A と B は IP アドレスとポート番号を表したものであり，S_A と S_B はシーケンス番号を形式化したものである．

ホスト A とホスト B は，各々プロセス P_A，P_B として形式化される．

$$\begin{aligned} P_A \stackrel{def}{=}\ & \mathsf{new}(S_A); \\ & \mathsf{store}\ x_{sa} = S_A; \\ & \mathsf{out}\ c\ \langle (A, B, x_{sa}) \rangle; \\ & \mathsf{inp}\ c\ (p); \\ & \mathsf{split}\ [x'_b, x'_a, x'_{sb}, x'_{sa1}]\ \mathsf{is}\ p\ \mathsf{err}\{\mathsf{free}\ x_{sa}, p\}; \\ & \mathsf{free}\ p; \\ & \mathsf{match}\ x'_a\ \mathsf{is}\ A\ \mathsf{and}\ x'_b\ \mathsf{is}\ B \\ & \quad \mathsf{and}\ x'_{sa1}\ \mathsf{is}\ \mathsf{succ}(x_{sa}) \\ & \ \mathsf{err}\{\mathsf{free}\ x_{sa}, x'_b, x'_a, x'_{sb}, x'_{sa1}, p\}; \\ & \mathsf{free}\ x'_b, x'_a, x'_{sa1}; \\ & \mathsf{out}\ c\ \langle (A, B, \mathsf{succ}(x_{sa}), \mathsf{succ}(x'_{sb})) \rangle; \\ & P'_A, \end{aligned}$$

$$\begin{aligned} P_B \stackrel{def}{=}\ & \mathsf{new}(S_B); \\ & \mathsf{inp}\ c\ (q_1); \\ & \mathsf{split}\ [y_a, y_b, y_{sa}]\ \mathsf{is}\ q_1\ \mathsf{err}\{\mathsf{free}\ q_1\}; \\ & \mathsf{free}\ q_1; \\ & \mathsf{match}\ y_b\ \mathsf{is}\ B\ \mathsf{err}\{\mathsf{free}\ y_a, y_b, y_{sa}\}; \\ & \mathsf{free}\ y_b; \\ & \mathsf{store}\ y_{sb} = S_B; \\ & \mathsf{out}\ c\ \langle (B, y_a, y_{sb}, \mathsf{succ}(y_{sa})) \rangle; \\ & \mathsf{inp}\ c\ (q_2); \\ & \mathsf{split}\ [y'_a, y'_b, y'_{sa1}, y'_{sb1}]\ \mathsf{is}\ q_2 \\ & \quad \mathsf{err}\{\mathsf{free}\ y_a, y_{sa}, y_{sb}, q_1, q_2\}; \\ & \mathsf{free}\ q_2; \\ & \mathsf{match}\ y'_b\ \mathsf{is}\ B\ \mathsf{and} \\ & \ y'_a\ \mathsf{is}\ y_a\ \mathsf{and} \\ & \ y'_{sa1}\ \mathsf{is}\ \mathsf{succ}(y_{sa})\ \mathsf{and} \\ & \ y'_{sb1}\ \mathsf{is}\ \mathsf{succ}(y_{sb}) \\ & \ \mathsf{err}\{\mathsf{free}\ y_a, y_{sa}, y_{sb}, y'_b, y'_a, y'_{sa1}, y'_{sb1}, q_1, q_2\}; \\ & \mathsf{free}\ y'_b, y'_a, y'_{sa1}, y'_{sb1}; \\ & P'_B \end{aligned}$$

通常の通信がおこなわれる場合のシステムは，以下のプロセス $NormalConfig$ により記述される．

$$NormalConfig \stackrel{def}{=} (\mathsf{repeat}\ P_A \mid \mathsf{repeat}\ P_B).$$

そして，プロセス P_A,P_B が，各々計算機 $\mathbf{a}$,$\mathbf{b}$ 上で実行されるとすると，このプロセス $NormalConfig$ は，$NormalConfig : (\mathbf{a} \mid \mathbf{b})$ と型付けされる．

前述の性質により初めの時刻はどこからはじめてもよいので $(0,0,0)$ とする．

$$\begin{aligned} & NormalConfig \\ \equiv\ & (\mathsf{repeat}\ P_A \mid \mathsf{repeat}\ P_B) \\ \xrightarrow{\tau}\ & \mathsf{new}(S_A); \mathsf{new}(S_B); \\ & (\mathsf{inp}\ c\ (p); \cdots) \mid \mathsf{repeat}\ P_A \\ & \mid (\mathsf{split}\ [y_a, y_b, y_c]\ \mathsf{is}\ q_1; \cdots) \mid \mathsf{repeat}\ P_B \\ & : \{\mathbf{a} \cdot store^{(0,0,1)}, \mathbf{b} \cdot store^{(0,1,0)}\} \\ \gg\ & \mathsf{new}(S_A); \mathsf{new}(S_B); \\ & ((\mathsf{inp}\ c\ (p); \cdots) \mid \mathsf{repeat}\ P_A \\ & \mid (\mathsf{out}\ c\ \langle (y_b, y_a, y_{sb}, \mathsf{succ}(y_{sa})) \rangle; \cdots) \\ & \mid \mathsf{repeat}\ P_B) \\ & : \{\mathbf{b} \cdot 3store^{(0,1,2)}\} - \{\mathbf{b} \cdot store^{(0,1,3)}\} \\ & + \{\mathbf{b} \cdot match^{(0,1,4)}\} \\ & \ + \{\mathbf{b} \cdot -store^{(0,1,5)}\} + \{\mathbf{b} \cdot store^{(0,1,6)}\} \\ & \ = \{\mathbf{b} \cdot 3store^{(0,1,2)}, \mathbf{b} \cdot -store^{(0,1,3)}, \\ & \ \mathbf{b} \cdot match^{(0,1,4)}, \mathbf{b} \cdot -store^{(0,1,5)}, \\ & \ \mathbf{b} \cdot store^{(0,1,6)}\} \\ \xrightarrow{\tau}\ & \mathsf{new}(S_A); \mathsf{new}(S_B); \\ & (\mathsf{split}\ [x'_b, x'_a, x'_{sb}, x'_{sa1}]\ \mathsf{is}\ p \cdots) \mid \mathsf{repeat}\ P_A \\ & \mid (\mathsf{inp}\ c\ (q_2); \cdots) \mid \mathsf{repeat}\ P_B \\ & : \{\mathbf{a} \cdot store^{(0,2,1)}\} \\ \gg\ & \mathsf{new}(S_A); \mathsf{new}(S_B); \\ & (\mathsf{out}\ c\ \langle (x_a, x_b, \mathsf{succ}(x_{sa}), \mathsf{succ}(x_{sb})) \rangle; \cdots) \end{aligned}$$

$$
\begin{aligned}
&\mid \mathsf{repeat}\ P_A \mid (\mathsf{inp}\ c\ (q_2); \cdots) \mid \mathsf{repeat}\ P_B \\
&: \{\mathbf{a} \cdot 4store^{(0,2,7)}\} + \{\mathbf{a} \cdot -store^{(0,2,8)}\} \\
&\quad + \{\mathbf{a} \cdot 3match^{(0,2,9)}\} - \{\mathbf{a} \cdot 3store^{(0,2,10)}\} \\
&\quad = \{\mathbf{a} \cdot 4store^{(0,2,7)}, \mathbf{a} \cdot -store^{(0,2,8)}, \\
&\qquad \mathbf{a} \cdot 3match^{(0,2,9)}, \mathbf{a} \cdot -3store^{(0,2,10)}\} \\
\xrightarrow{\tau}\ & \mathsf{new}(S_A); \mathsf{new}(S_B); \\
& (P_A \mid \mathsf{repeat}\ P_A \\
&\quad \mid (\mathsf{split}\ [y'_a, y'_b, y'_{sa1}, y'_{sb1}]\ \mathsf{is}\ q_2 \cdots) \\
&\quad \mid \mathsf{repeat}\ P_B) \\
&: \{\mathbf{b} \cdot store^{(0,3,6)}\} \\
\gg\ & \mathsf{new}(S_A); \mathsf{new}(S_B); \\
& P'_A \mid \mathsf{repeat}\ P_A \mid P'_B \mid \mathsf{repeat}\ P_B \\
&: \{\mathbf{b} \cdot 4store^{(0,3,11)}\} + \{\mathbf{b} \cdot -store^{(0,3,12)}\} \\
&\quad + \{\mathbf{b} \cdot 4match^{(0,3,13)}\} + \{\mathbf{b} \cdot -4store^{(0,3,14)}\} \\
&\quad = \{\mathbf{b} \cdot 4store^{(0,3,11)}, \mathbf{b} \cdot -store^{(0,3,12)}, \\
&\qquad \mathbf{b} \cdot 4match^{(0,3,13)}, \mathbf{b} \cdot -4store^{(0,3,14)}\}
\end{aligned}
$$

結局，通常の 3 ウェイハンドシェイクの場合，以下のようにコストが費やされることがわかる．

$$
\begin{aligned}
& NormalConfig \\
\longrightarrow\!\!\!\!\rightarrow\ & (\mathsf{new}(S_A); P'_A) \mid \mathsf{repeat}\ P_A \\
& \mid (\mathsf{new}(S_B); P'_B) \mid \mathsf{repeat}\ P_B \\
& : \{\mathbf{a} \cdot store^{(0,0,1)}, \mathbf{a} \cdot store^{(0,2,1)}, \\
& \quad \mathbf{a} \cdot 4store^{(0,2,7)}, \mathbf{a} \cdot -store^{(0,2,8)}, \\
& \quad \mathbf{a} \cdot 3match^{(0,2,9)}, \mathbf{a} \cdot -3store^{(0,2,10)}, \\
& \quad \mathbf{b} \cdot store^{(0,1,0)}, \mathbf{b} \cdot 3store^{(0,1,2)}, \\
& \quad \mathbf{b} \cdot -store^{(0,1,3)}, \mathbf{b} \cdot match^{(0,1,4)}, \\
& \quad \mathbf{b} \cdot -store^{(0,1,5)}, \mathbf{b} \cdot store^{(0,1,6)}, \\
& \quad \mathbf{b} \cdot store^{(0,3,6)}, \mathbf{b} \cdot 4store^{(0,3,11)}, \\
& \quad \mathbf{b} \cdot -store^{(0,3,12)}, \mathbf{b} \cdot 4match^{(0,3,13)}, \\
& \quad \mathbf{b} \cdot -4store^{(0,3,14)}\}
\end{aligned}
$$

次に SYN あふれ攻撃の例をみてゆこう．B を攻撃する攻撃者を I とする．

$$
\begin{aligned}
I \to B &: I_i, B, S_{I_i}, \\
B \to I &: B, I_i, S_B, S_{I_i} + 1, \\
&\quad (i = 1, 2, \ldots)
\end{aligned}
$$

攻撃者 I は以下のプロセス P_I として定式化される．

$$P_I \stackrel{def}{=} \mathsf{new}(i); \mathsf{new}(s); \mathsf{out}\ c\ \langle (i, B, s) \rangle; \mathsf{stop}.$$

被害者 B から攻撃者 I に送られるメッセージは，IP アドレスが擬装されるため，誰にも受信されることなくネットワークにおいて失なわれる．このようなメッセージの消失を形式化するために，ネットワークを表現するプロセス P_N を設ける．

$$P_N \stackrel{def}{=} \mathsf{inp}\ c\ (r); \mathsf{stop}.$$

攻撃者を表すプロセス P_I に計算機の型 $\mathbf{i}$ を割り当て，ネットワークを表すプロセス P_I に型 $\mathbf{n}$ を割り当て，被攻撃者となるプロセス P_B に型 $\mathbf{b}$ を割り当てるとする．このときの状況を表現するプロセス $AttackConfig$ は以下のとおり．

$$AttackConfig \stackrel{def}{=} (\mathsf{repeat}\ P_I \mid \mathsf{repeat}\ P_B \mid \mathsf{repeat}\ P_N).$$

このプロセスは以下のようなコミットメント関係・簡約関係の列をもつ．

$$
\begin{aligned}
& AttackConfig \\
\equiv\ & (\mathsf{repeat}\ P_I \mid \mathsf{repeat}\ P_B \\
& \quad \mid \mathsf{repeat}\ P_N) \\
\xrightarrow{\tau}\ & \mathsf{new}(S_B); \\
& (\mathsf{stop} \mid \mathsf{repeat}\ P_I \\
& \quad \mid (\mathsf{split}\ [y_a, y_b, y_c]\ \mathsf{is}\ q_1; \cdots) \\
& \quad \mid \mathsf{repeat}\ P_B \\
& \quad \mid \mathsf{inp}\ c\ (r); \mathsf{stop} \mid \mathsf{repeat}\ P_N) \\
& : \{\mathbf{b} \cdot store^{(0,1,0)}\} \\
\gg\ & \mathsf{new}(S_B); \\
& (\mathsf{stop} \mid \mathsf{repeat}\ P_I \\
& \quad \mid \mathsf{out}\ c\ \langle (B, A, S_B, \mathsf{succ}(y_{sa})) \rangle; \cdots \\
& \quad \mid \mathsf{repeat}\ P_B \\
& \quad \mid \mathsf{inp}\ c\ (r); \mathsf{stop} \mid \mathsf{repeat}\ P_N) \\
& : \{\mathbf{b} \cdot 3store^{(0,1,1)}\} + \{\mathbf{b} \cdot -store^{(0,1,2)}\} \\
& + \{\mathbf{b} \cdot match^{(0,1,3)}\} + \{\mathbf{b} \cdot -store^{(0,1,4)}\} \\
& + \{\mathbf{b} \cdot store^{(0,1,5)}\} \\
\gg\ & \mathsf{new}(S_B); \\
& (\mathsf{stop} \mid \mathsf{repeat}\ P_I \mid \mathsf{inp}\ c\ (q_2); \cdots \\
& \quad \mid \mathsf{repeat}\ P_B \mid \mathsf{stop} \mid \mathsf{repeat}\ P_N) \\
& : \{\mathbf{n} \cdot store^{(0,2,0)}\}
\end{aligned}
$$

したがって，この場合のコストは以下のようになる．

$$AttackConfig$$

$$
\begin{aligned}
\longrightarrow\!\!\!\!\rightarrow \tau \quad & \mathsf{new}(S_B); \\
& (\mathsf{stop} \mid \mathsf{repeat}\, P_I \mid \mathsf{inp}\ c\ (q_2); \cdots \\
& \mid \mathsf{repeat}\, P_B \mid \mathsf{stop} \mid \mathsf{repeat}\, P_N) \\
& : \{\mathbf{b} \cdot store^{(0,1,0)}, \mathbf{b} \cdot 3store^{(0,1,1)}, \\
& \mathbf{b} \cdot -store^{(0,1,2)}, \mathbf{b} \cdot match^{(0,1,3)}, \\
& \mathbf{b} \cdot -store^{(0,1,4)}, \mathbf{b} \cdot store^{(0,1,5)}, \\
& \mathbf{n} \cdot store^{(0,2,0)}\}
\end{aligned}
$$

これらのコスト解析は，論文 [TNI04, 冨池西] で与えたものとも同じである．主な異なる点は，わかったコスト解析結果において，コストの発生時刻が明示されているところである．

4 おわりに

本論文では，プロトコルにおけるサービス不能攻撃耐性を解析するための計算体系である Spice 計算を拡張し，各コストの発生時刻を明示的に表現されるようにした時刻つき Spice 計算を提唱した．この時刻つき Spice 計算では，階層化した時間モデルに基づき，操作的意味論である遷移関係において計算ステップの前後で時刻の推移を組み込まれた．

参考文献

[AG97a] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communication Security*, pages 36–47. ACM Press, 1997.

[AG97b] M. Abadi and A. D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *CONCUR'97: Concurrency Theory*, volume 1243, pages 59–73. Springer-Verlag, Berlin Germany, 1997.

[Mea99] Catherine Meadows. A formal framework and evaluation method for network denial of service. In *Proceeding of the 12th IEEE Computer Security Foundations Workshop*, pages 4–13, 1999.

[SKK+97] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, and Diego Zamboni. Analysis of a denial of service attack on TCP. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 208–223. IEEE Computer Society, IEEE Computer Society Press, May 1997.

[TNI04] Daigo Tomioka, Shin-ya Nishizaki, and Ritsuya Ikeda. A cost estimation calculus for analyzing the resistance to denial-of-service attack. In *Software Security – Theories and Systems.*, volume 3233 of *Lecture Notes in Computer Science*, pages 25–44, 2004.

[冨池西] 冨岡大悟, 池田立野, and 西崎真也. 通信プロトコルにおけるサービス不能攻撃耐性解析のための型付き π 計算. コンピュータソフトウェア誌に掲載予定.