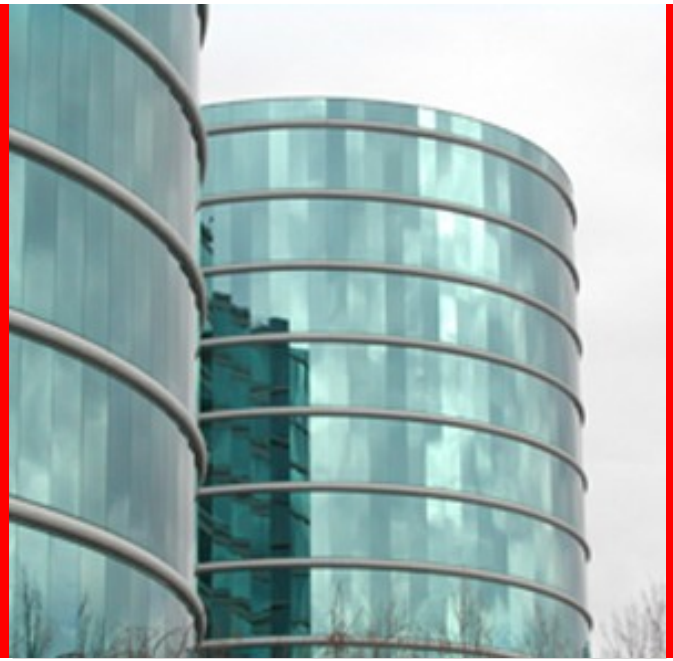


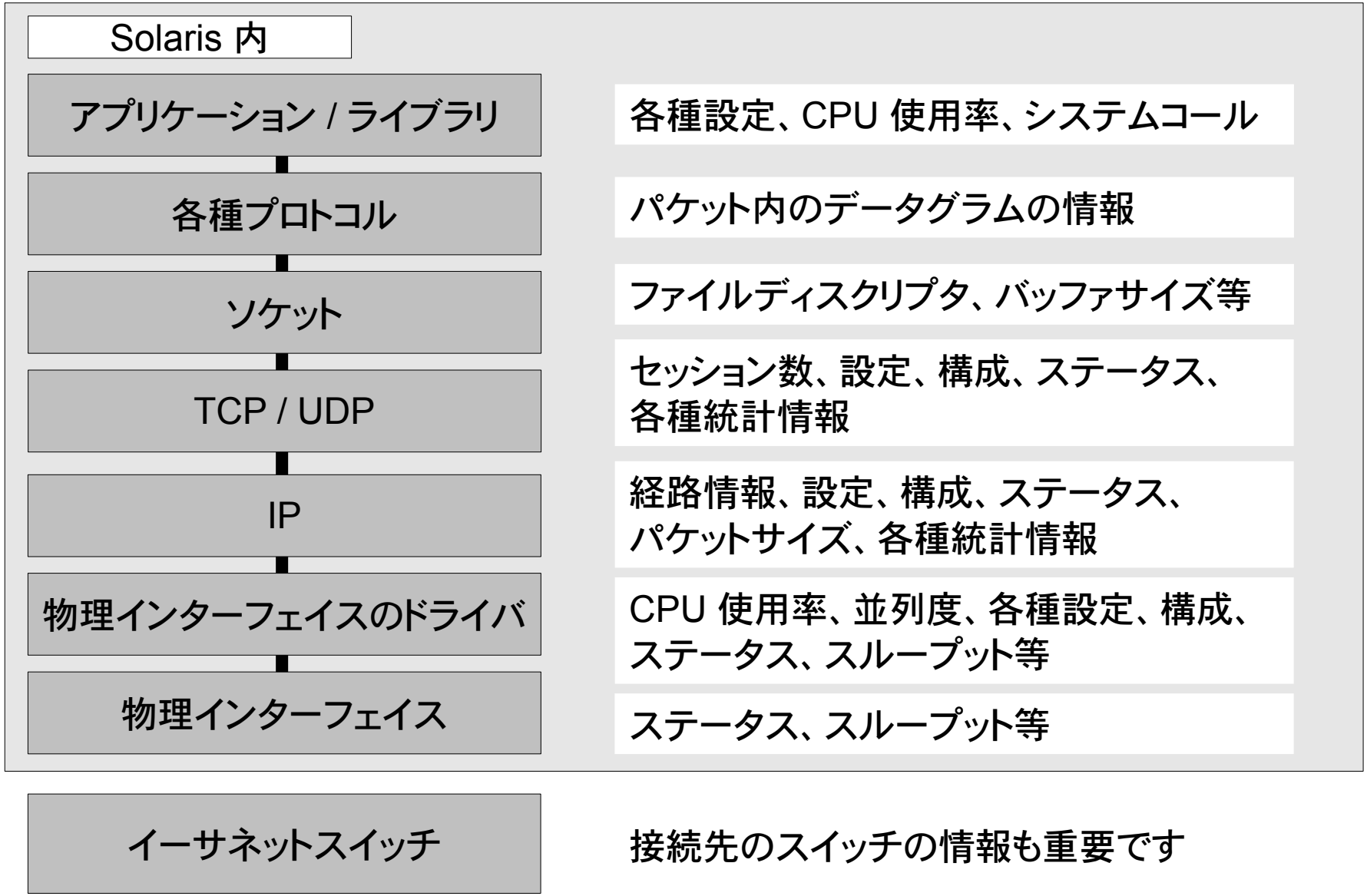
ORACLE®

# Solaris: ネットワークの解析コマンド入門

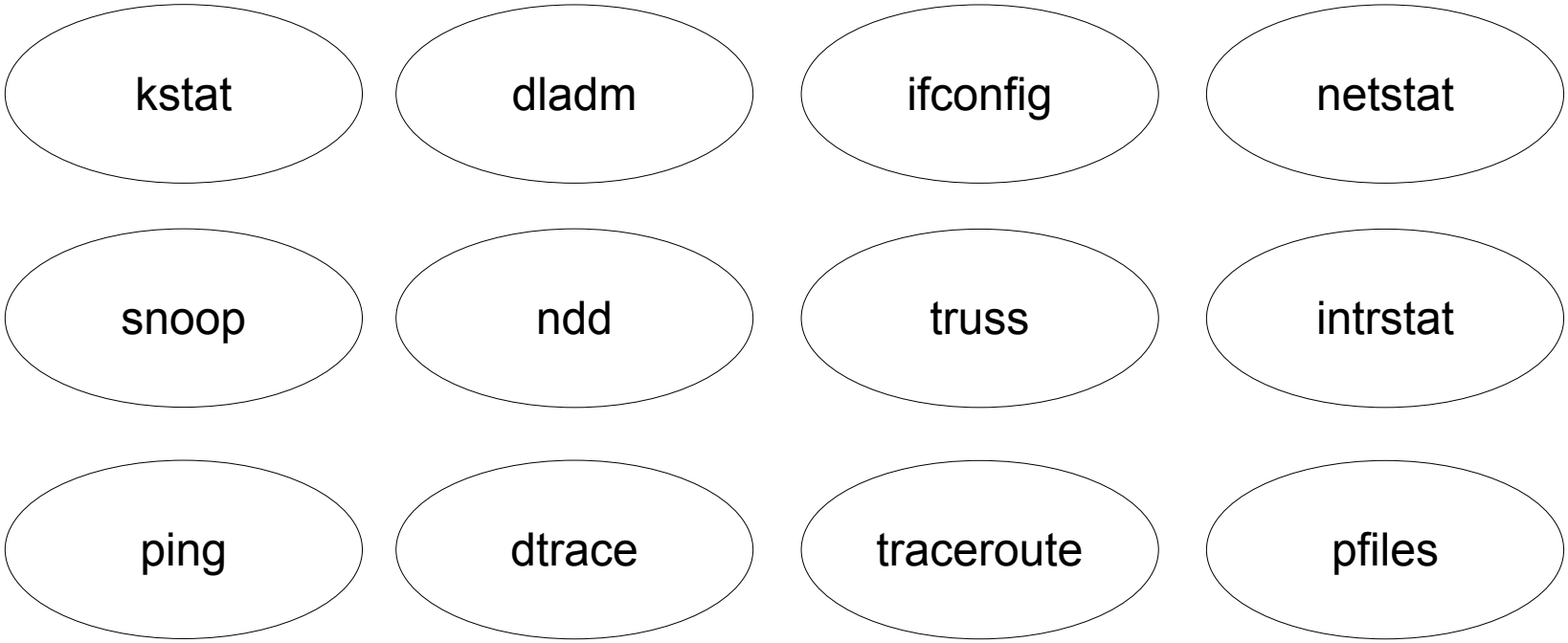
日本オラクル株式会社 システム事業統括
セールスコンサルタント 本間大輔

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

# ネットワークに関する主な解析ポイント

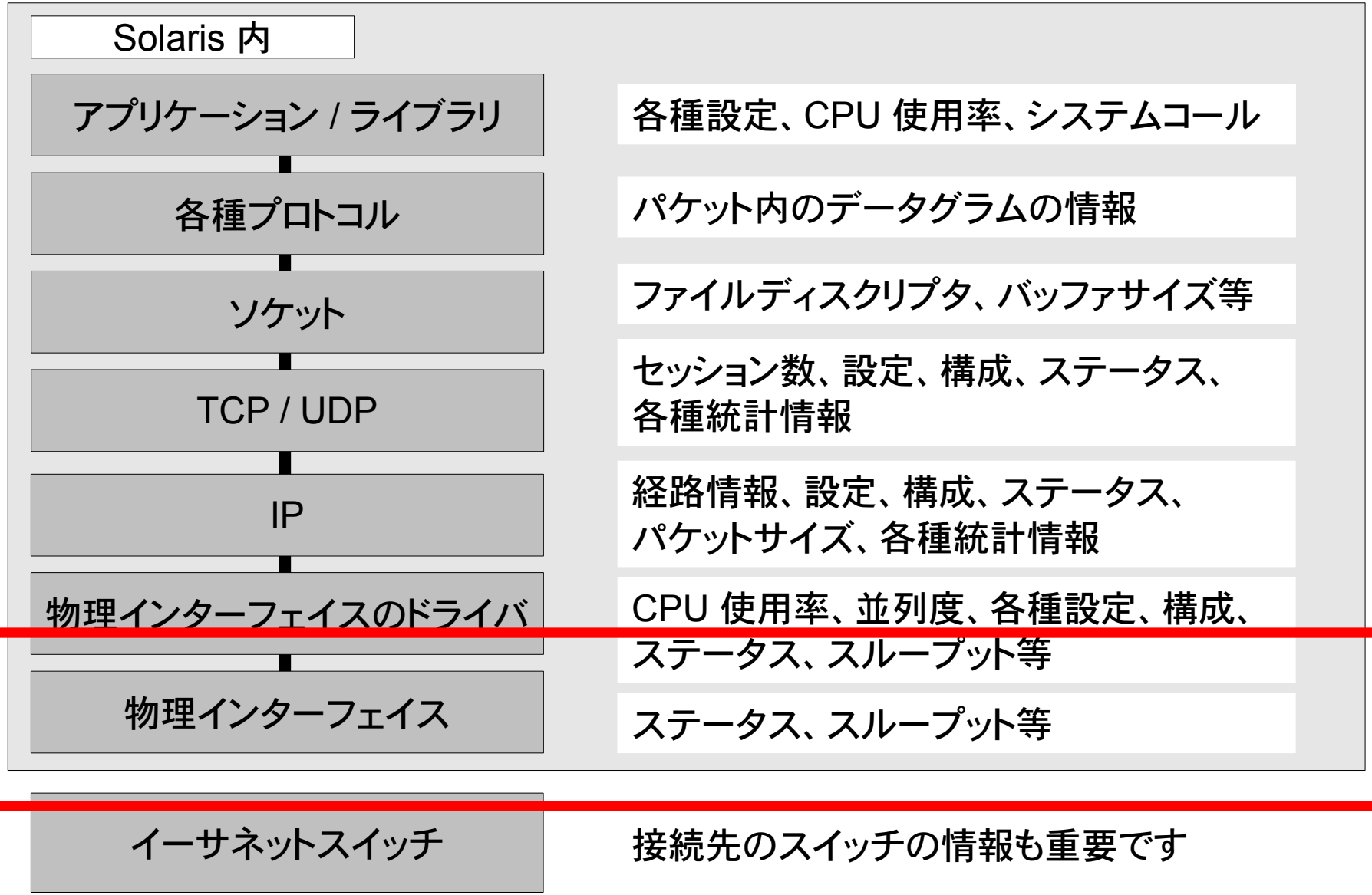


# 主に使用するコマンド




~~ これらを押さえておけば大体大丈夫です ~~

# まずは物理インターフェイスから



# Tips #1

## マシンに存在する全ての
## ネットワークインターフェイスを
## 一覧表示する方法

~~ まずは調査対象となる機材の
物理構成を確認しましょう ~~

ORACLE

# Tips #1 NIC の一覧表示

- "ifconfig -a" でインターフェイスの一覧を表示します

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.16.67.5 netmask ffff0000 broadcast 10.16.255.255
        ether 0:23:8b:64:88:60
```


Solaris のインターフェイス名は**ドライバ名 + 通し番号**の形式で振られます

~~ ifconfig コマンドはネットワークインターフェイスを
コンフィグ（構成）するコマンドです ~~

# Tips #1 NIC の一覧表示

- "ifconfig -a" でインターフェイスの一覧を表示します

```
# ifconfig -a          ←―― インターフェイスの一覧を表示
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.16.67.5 netmask ffff0000 broadcast 10.16.255.255
        ether 0:23:8b:64:88:60
```

ループバックインターフェイス

物理インターフェイス

- ただし、これでは ifconfig <NIC> plumb で有効にした NIC しか表示されません。そこで、、、

~~ ifconfig コマンドはネットワークインターフェイスを
コンフィグ（構成）するコマンドです ~~

# Tips #1 NIC の一覧表示

- "ifconfig -a plumb" で全てのインターフェイスを有効化し、
- "ifconfig -a" でインターフェイスの一覧を表示します

```
# ifconfig -a plumb ←―― 全てのインターフェイスを有効化
ifconfig: SIOCSLIFNAME for ip: e1000g0: already exists
# ifconfig -a ←―― インターフェイスの一覧を表示
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.16.67.5 netmask ffff0000 broadcast 10.16.255.255
        ether 0:23:8b:64:88:60
e1000g1: flags=201000842<BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 3
        inet 0.0.0.0 netmask 0
        ether 0:23:8b:64:88:61
e1000g2: flags=201000842<BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 4
        inet 0.0.0.0 netmask 0
        ether 0:23:8b:64:88:62
e1000g3: flags=201000842<BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 5
        inet 0.0.0.0 netmask 0
        ether 0:23:8b:64:88:63
```

~~ ifconfig -a plumb で全てのインターフェイスを plumb する事が出来ます ~~

# Tips #1 NIC の一覧表示

- "ifconfig -a plumb" で全てのインターフェイスを有効化し、
- "ifconfig -a" でインターフェイスの一覧を表示します

```
# ifconfig -a plumb
ifconfig: SIOCSLIFNAME for ip: e1000g0: already exists
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.16.67.5 netmask ffff0000 broadcast 10.16.255.255
        ether 0:23:8b:64:88:60
e1000g1: flags=201000842<BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 3
        inet 0.0.0.0 netmask 0
        ether 0:23:8b:64:88:61
e1000g2: flags=201000842<BROADC
        inet 0.0.0.0 netmask 0
        ether 0:23:8b:64:88:62
e1000g3: flags=201000842<BROADC
        inet 0.0.0.0 netmask 0
        ether 0:23:8b:64:88:63
```

全てのインターフェイスを有効化

インターフェイスの一覧を表示

ループバックインターフェイス

物理インターフェイス x4

~~ ifconfig -a plumb で全てのインターフェイスを plumb する事が出来ます ~~

# 注意!

- 環境構築時やテスト環境では ifconfig -a plumb は手軽でとても便利です。
- ただ、サービスインしている環境で不用意にシステム構成を変更する事は**危険**ですので、ifconfig -a plumb は注意して使用して下さい。
- サービスインしている環境で、plumb されていないインターフェイスも含め、全てのインターフェイスを一覧表示したい場合は、次の dladm コマンドや kstat コマンドを使用します。

# Tips #1 NIC の一覧表示

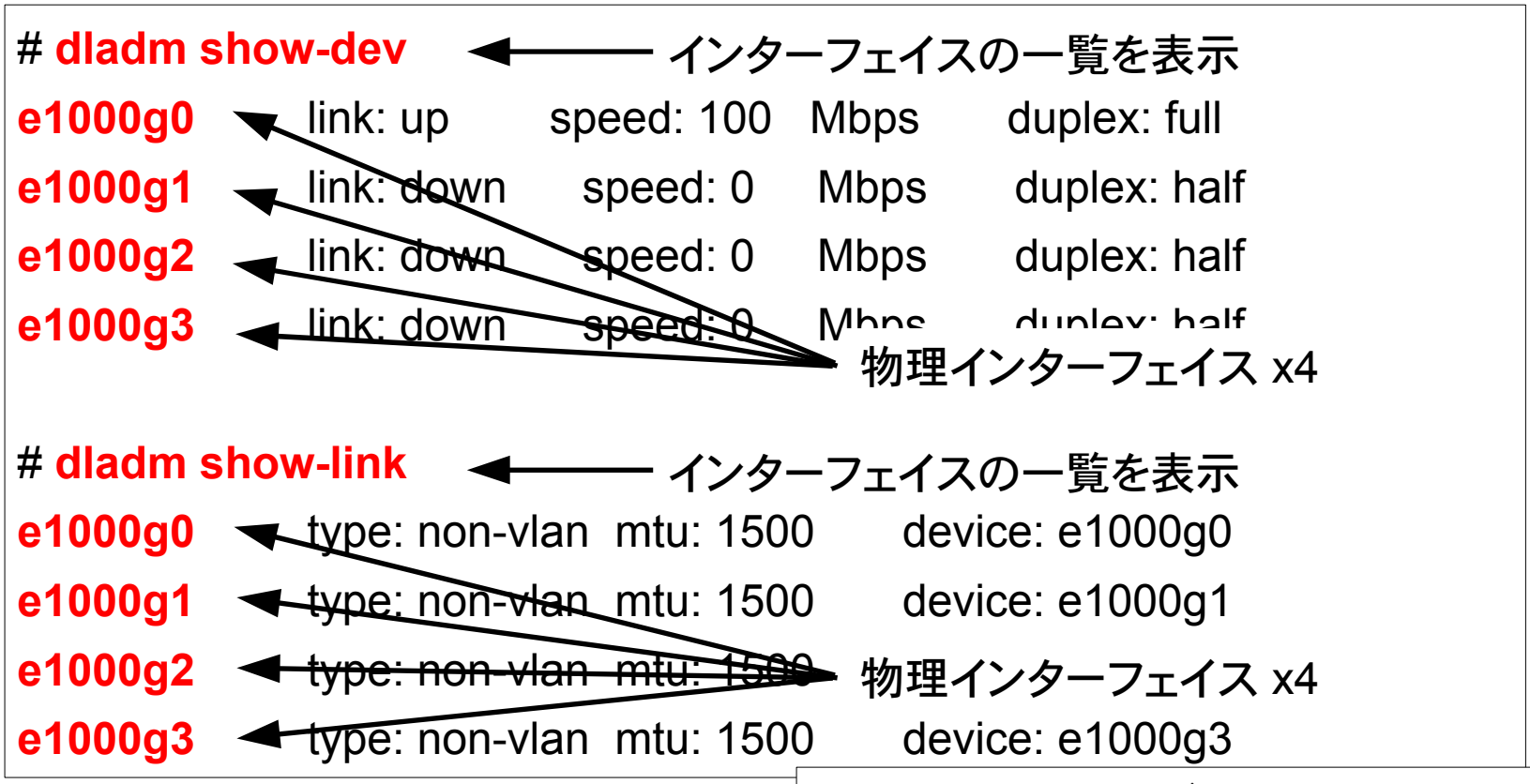
- "dladm show-dev" 及び "dladm show-link" コマンドでインターフェイスの一覧を表示します

```
# dladm show-dev      ←――― インターフェイスの一覧を表示
e1000g0         link: up        speed: 100   Mbps       duplex: full
e1000g1         link: down      speed: 0     Mbps       duplex: half
e1000g2         link: down      speed: 0     Mbps       duplex: half
e1000g3         link: down      speed: 0     Mbps       duplex: half

# dladm show-link     ←――― インターフェイスの一覧を表示
e1000g0         type: non-vlan  mtu: 1500       device: e1000g0
e1000g1         type: non-vlan  mtu: 1500       device: e1000g1
e1000g2         type: non-vlan  mtu: 1500       device: e1000g2
e1000g3         type: non-vlan  mtu: 1500       device: e1000g3
```

~~ dladm コマンドはデータリンクレイヤーを
管理するコマンドです ~~

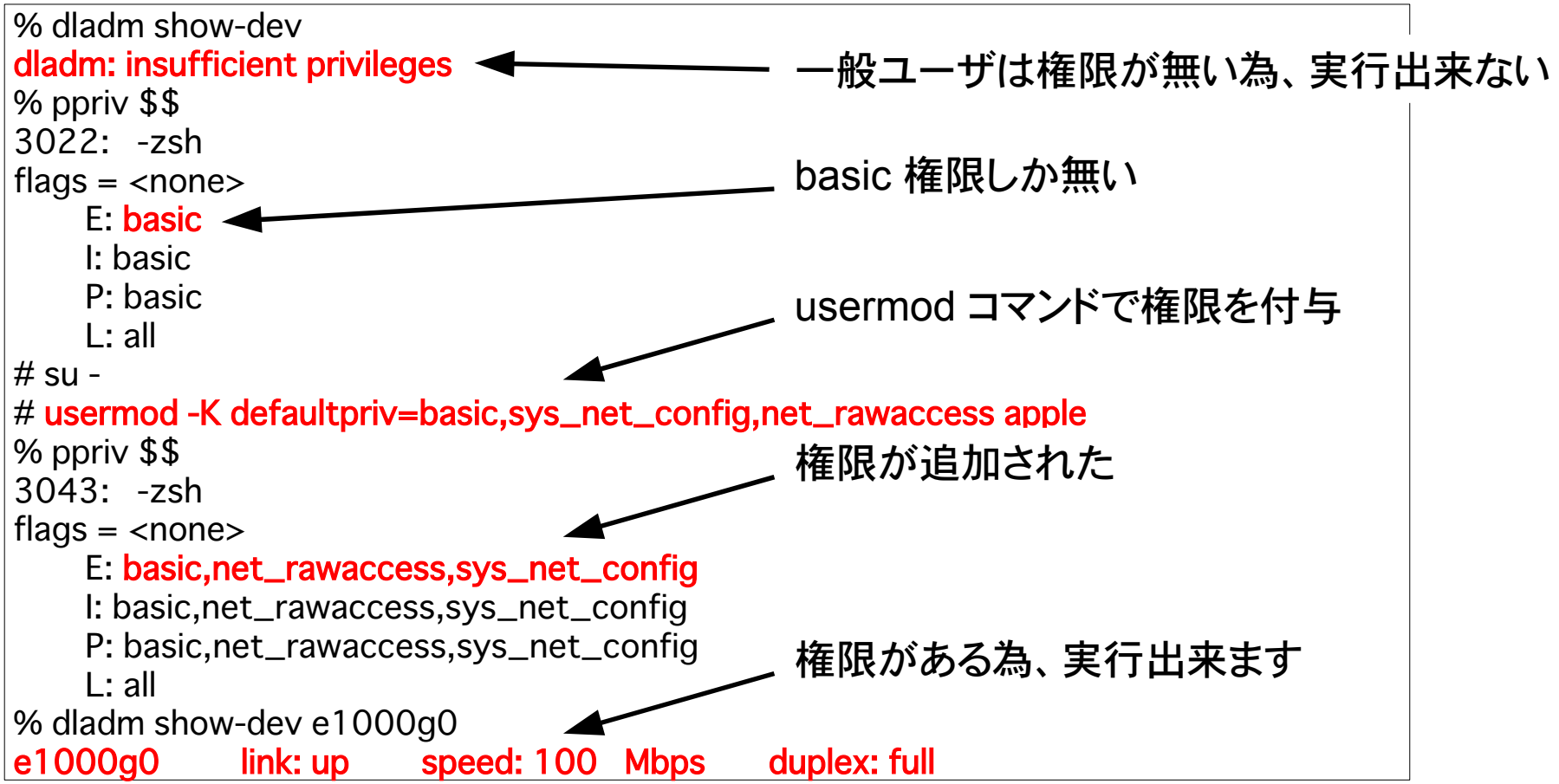
# Tips #1 NIC の一覧表示

- "dladm show-dev" 及び "dladm show-link" コマンドでインターフェイスの一覧を表示します



# 注意!

- Solaris 10 では、dladm コマンドの実行は root ユーザになるか sys_net_config, net_rawaccess 権限が必要です



# Tips #1 NIC の一覧表示

- "kstat -p ':::link_up'" コマンドでインターフェイスの一覧を表示します
- kstat の実行には特別な権限は必要なく、plumb されていないインターフェイスも調査可能です

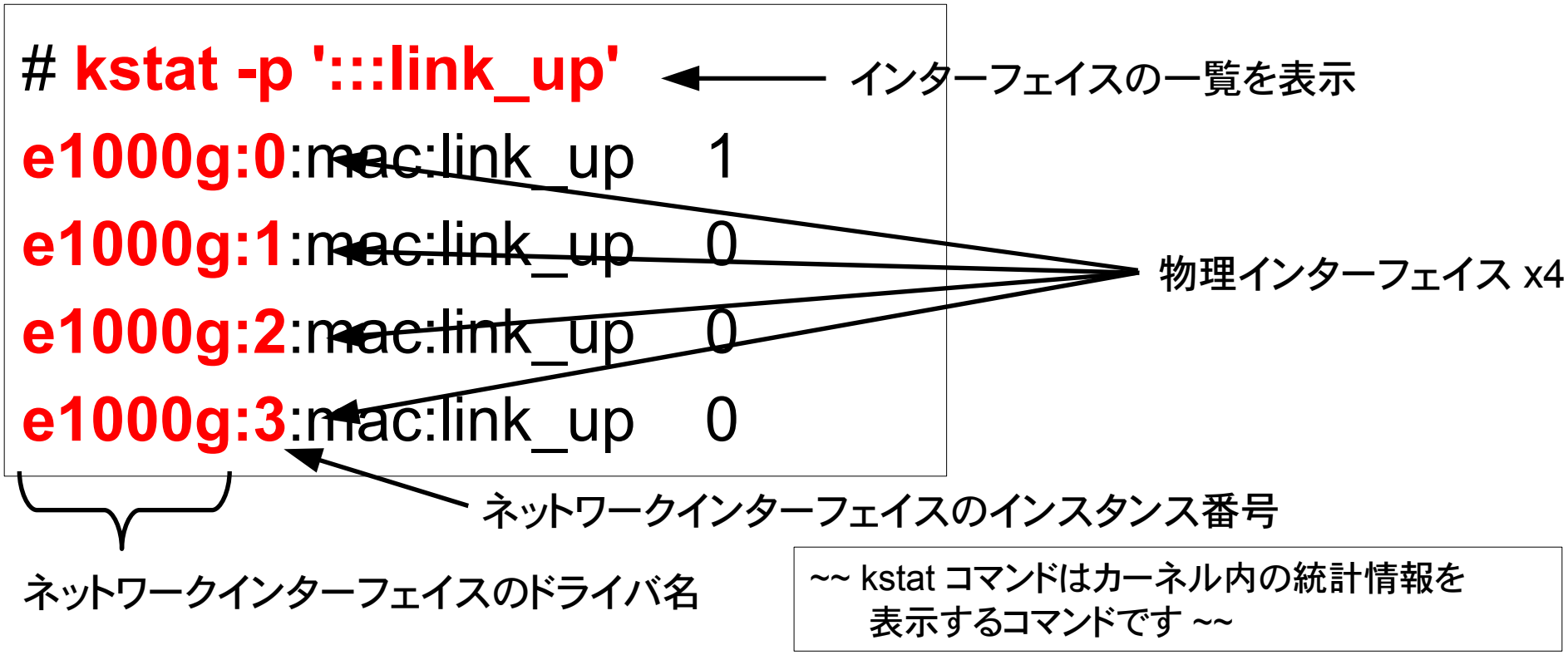
```
# kstat -p ':::link_up'
e1000g:0:mac:link_up    1
e1000g:1:mac:link_up    0
e1000g:2:mac:link_up    0
e1000g:3:mac:link_up    0
```

← インターフェイスの一覧を表示

~~ kstat コマンドはカーネル内の統計情報を
表示するコマンドです ~~

# Tips #1 NIC の一覧表示

- "kstat -p ':::link_up'" コマンドでインターフェイスの一覧を表示します
- kstat の実行には特別な権限は必要なく、plumb されていないインターフェイスも調査可能です



# Tips #1 NIC の一覧表示

- ifconfig コマンドは論理インターフェイスを表示する事も可能です

```
# ifconfig e1000g0 addif 10.16.67.205/16 broadcast + up
Created new logical interface e1000g0:1
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.16.67.5 netmask ffff0000 broadcast 10.16.255.255
        ether 0:23:8b:64:88:60
e1000g0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.16.67.205 netmask ffff0000 broadcast 10.16.255.255
```

論理インターフェイスの作成

物理インターフェイス

論理インターフェイス

~~ 論理インターフェイスを調べたい場合は
ifconfig コマンドを使用します ~~

# Tips #1 NIC の一覧表示

- dladm コマンドは簡単にスクリプトと組み合わせられます
- NIC=`dladm show-dev -p | awk '{printf "%s ", $1}'` コマンドで ${NIC} 変数にインターフェイスの一覧を格納

```
# NIC=`dladm show-dev -p | awk '{printf "%s ", $1}'`
# echo ${NIC}
e1000g0 e1000g1 e1000g2 e1000g3
```

インターフェイスの一覧を格納

物理インターフェイス x4

- リンクアップしているインターフェイスのスループットをログに出力するスクリプト

```
#!/bin/sh
NIC=`dladm show-dev -p | grep 'link=up' | awk '{printf "%s ", $1}'`
for i in ${NIC}
  do dladm show-dev -s -i 1 ${i} > dladm-${i}.log &
done
```

~~ dladm コマンドは、この様にシェルスクリプト内から使用すると便利です ~~

# Tips #1 NIC の一覧表示

- 各コマンドの違い
  - ifconfig コマンドは論理インターフェイスを表示する事も可能です
  - dladm や kstat では論理インターフェイスを出力する事は出来ません
  - ifconfig コマンドは plumb していないインターフェイスを表示する事は出来ません
  - dladm や kstat コマンドは plumb していないインターフェイスも調査出来ます
  - dladm や kstat コマンドは、出力を簡単にパース出来るという特徴を活かして、スクリプトの中からインターフェイスの一覧を取得したい場合に便利です。
  - ifconfig 及び kstat コマンドはどのユーザでも実行可能です
  - dladm コマンドの実行は root ユーザになるか、sys_net_config 権限が必要です

# Tips #1 NIC の一覧表示

| | 論理インターフェイスを表示出来る | plumb されていないインターフェイスを表示出来る | スクリプトから使い易い | 特別な権限無しに実行出来る |
|---|---|---|---|---|
| ifconfig | yes | no | no | yes |
| dladm | no | yes | yes | no |
| kstat | no | yes | yes | yes |

# Tips #1

## マシンに存在する全ての
## ネットワークインターフェイスを
## 一覧表示する方法

```
# ifconfig -a plumb && ifconfig -a
# dladm show-dev
# dladm show-link
# kstat -p ':::link_up'
```

~~ どのコマンドも非常に頻繁に使用する
コマンドです。是非憶えて下さい。~~

ORACLE

# Tips #2

## ネットワークインターフェイスの
## ステータスを調べる方法

~~ インターフェイスがきちんと機能
している事を確認します ~~

ORACLE

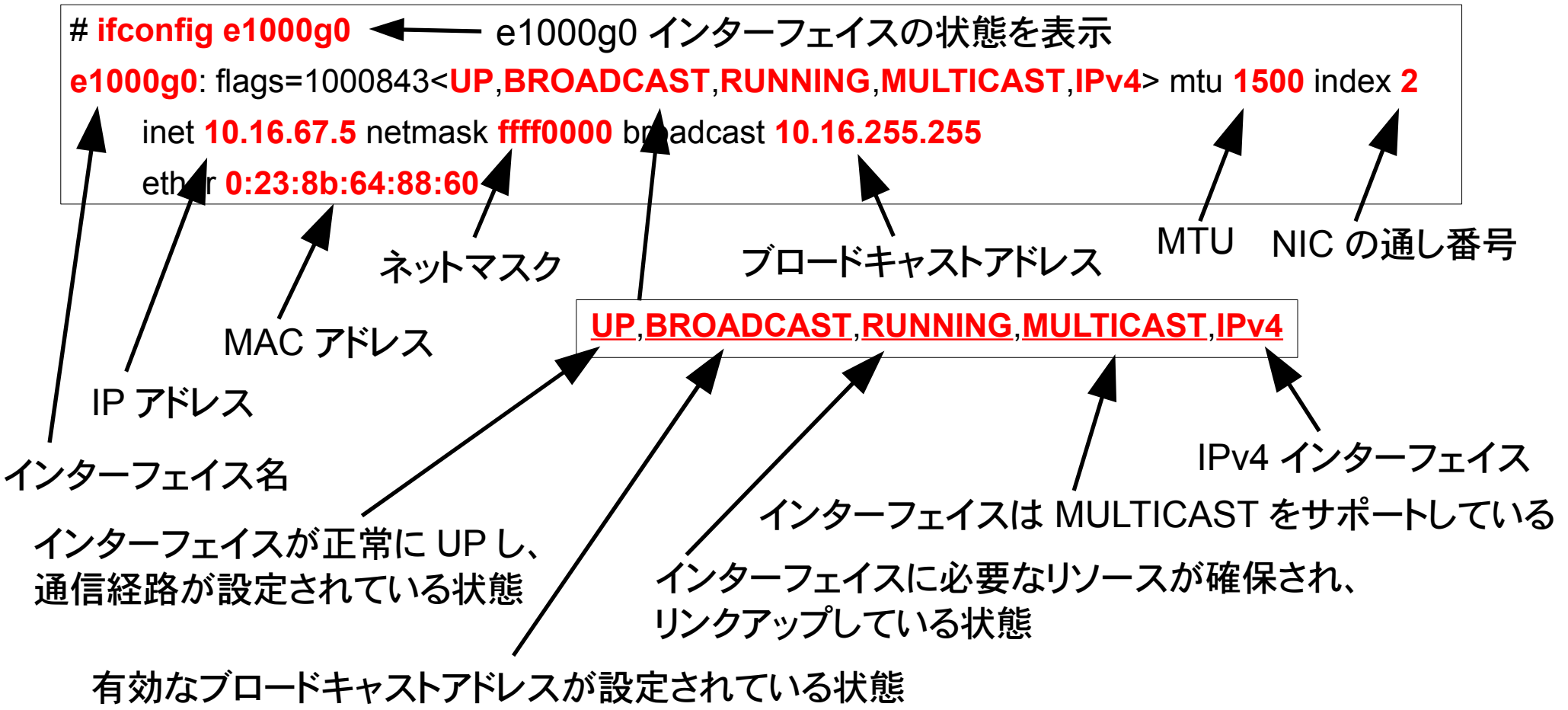
# Tips #2 NIC の状態の表示

- "ifconfig <NIC>" でインターフェイスの情報を表示出来ます
- root ユーザで実行した場合は MAC アドレスも出力されます

```
# ifconfig e1000g0  ◀―― e1000g0 インターフェイスの状態を表示
e1000g0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.16.67.5 netmask ffff0000 broadcast 10.16.255.255
        ether 0:23:8b:64:88:60
```

# Tips #2 NIC の状態の表示

- "ifconfig <NIC>" でインターフェイスの情報を表示出来ます
- root ユーザで実行した場合は MAC アドレスも出力されます



# Tips #2 NIC の状態の表示

- dladm コマンドでインターフェイスの情報を表示出来ます

```
# dladm show-dev      ←― インターフェイスの状態を表示
e1000g0       link: up       speed: 100   Mbps      duplex: full
e1000g1       link: down     speed: 0     Mbps      duplex: half
e1000g2       link: down     speed: 0     Mbps      duplex: half
e1000g3       link: down     speed: 0     Mbps      duplex: half
```

リンクステータス　接続速度　duplex 情報

```
# dladm show-link     ←― インターフェイスの状態を表示
e1000g0       type: non-vlan  mtu: 1500      device: e1000g0
e1000g1       type: non-vlan  mtu: 1500      device: e1000g1
e1000g2       type: non-vlan  mtu: 1500      device: e1000g2
e1000g3       type: non-vlan  mtu: 1500      device: e1000g3
```

VLAN 情報　MTU

# Tips #2 NIC の状態の表示

- kstat コマンドでインターフェイスの情報を表示出来ます

```
# kstat -p ':::link_up'      ←――― インターフェイスのリンクステータスを表示
e1000g:0:mac:link_up    1
e1000g:1:mac:link_up    0
e1000g:2:mac:link_up    0
e1000g:3:mac:link_up    0
# kstat -p ':::ifspeed'      ←――― インターフェイスの接続速度を表示
e1000g:0:e1000g0:ifspeed        100000000
e1000g:0:mac:ifspeed    100000000
e1000g:1:mac:ifspeed    0
e1000g:2:mac:ifspeed    0
e1000g:3:mac:ifspeed    0
# kstat -p ':::link_duplex'  ←――― インターフェイスの duplex 情報を表示
e1000g:0:mac:link_duplex        2
e1000g:1:mac:link_duplex        1
e1000g:2:mac:link_duplex        1
e1000g:3:mac:link_duplex        1
```

~~ パラメータ名と値はドライバの実装に依存します ~~

# Tips #2 NIC の状態の表示

- kstat コマンドでインターフェイスの情報を表示出来ます

```
# kstat -p ':::link_up'
e1000g:0:mac:link_up    1
e1000g:1:mac:link_up    0
e1000g:2:mac:link_up    0
e1000g:3:mac:link_up    0
# kstat -p ':::ifspeed'
e1000g:0:e1000g0:ifspeed        100000000
e1000g:0:mac:ifspeed    100000000
e1000g:1:mac:ifspeed    0
e1000g:2:mac:ifspeed    0
e1000g:3:mac:ifspeed    0
# kstat -p ':::link_duplex'
e1000g:0:mac:link_duplex        2
e1000g:1:mac:link_duplex        1
e1000g:2:mac:link_duplex        1
e1000g:3:mac:link_duplex        1
```

インターフェイスのリンクステータスを表示

1 は link up

0 は link down

インターフェイスの接続速度を表示

100000000bps => 100Mbps

link up していないので 0

インターフェイスの duplex 情報を表示

2 は full duplex

1 は half duplex

~~ パラメータ名と値はドライバの実装に依存します ~~

# Tips #2 NIC の状態の表示

- ndd コマンドでインターフェイスの情報を表示出来ます

```
# ndd -get /dev/e1000g0 link_status
1          ← 1 : link up
```
e1000g0 インターフェイスのリンクステータスを表示

```
# ndd -get /dev/e1000g0 link_speed
100        ← 100Mbps
```
e1000g0 インターフェイスの接続速度を表示

```
# ndd -get /dev/e1000g0 link_duplex
2          ← 2 : full duplex
```
e1000g0 インターフェイスの duplex 情報を表示

~~ パラメータ名と値はドライバの実装に依存します ~~

# Tips #2

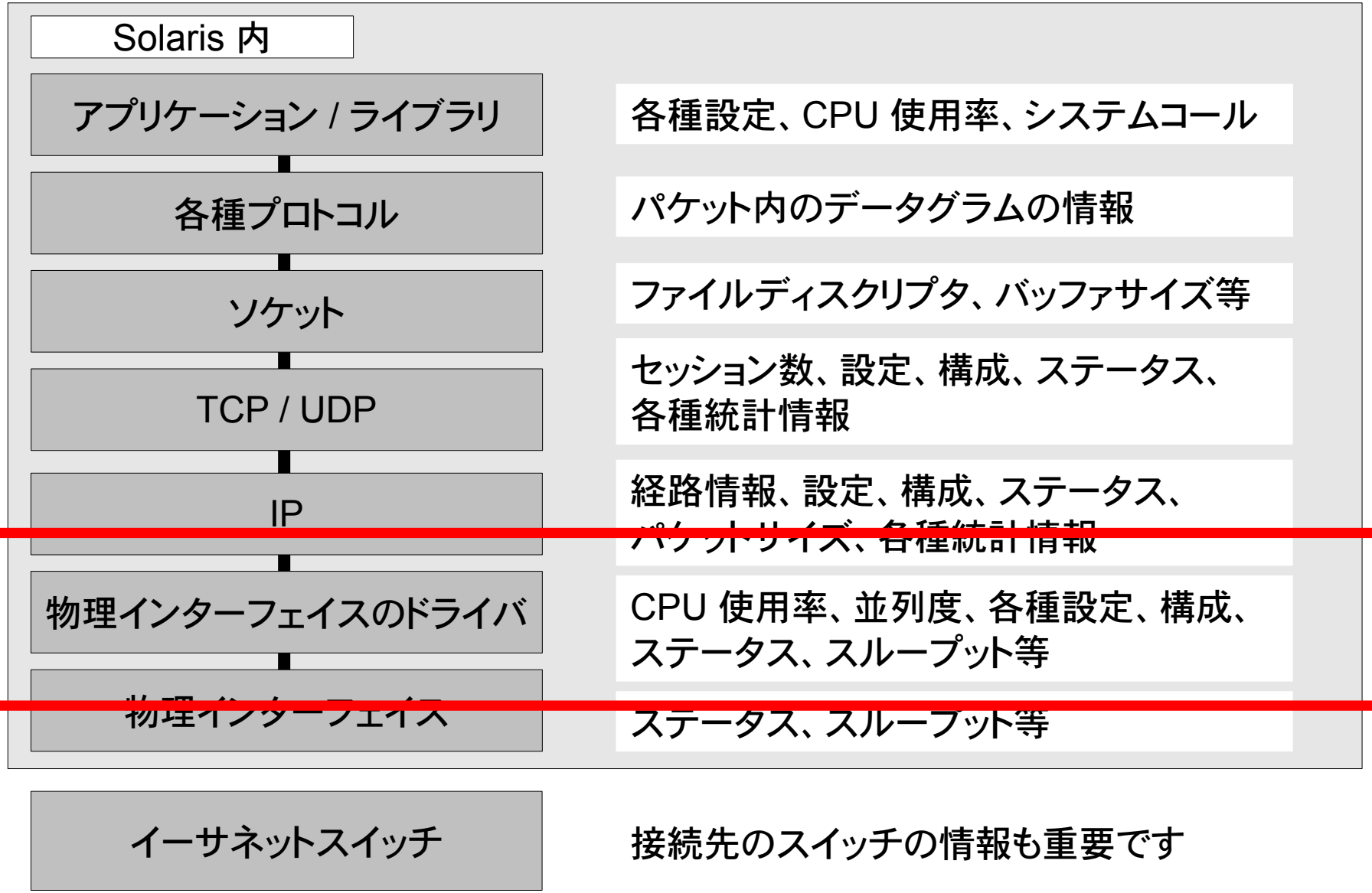
## ネットワークインターフェイスの ステータスを調べる方法

```
# ifconfig <NIC>
# dladm show-dev
# dladm show-link
# kstat -p ':::link_up'
# kstat -p ':::ifspeed'
# kstat -p ':::link_duplex'
# ndd -get /dev/e1000g0 link_status
# ndd -get /dev/e1000g0 link_speed
# ndd -get /dev/e1000g0 link_duplex
```

~~ kstat コマンドを使用すると、その他にも 様々なデータにアクセス出来ます ~~

ORACLE

# ここからは、一段上がってドライバのレイヤーです



# Tips #3

## ネットワークインターフェイス毎のスループットを調べる方法

~~ インターフェイスの性能に
問題が無いかを調べます ~~

ORACLE

# Tips #3 NIC のスループットの表示

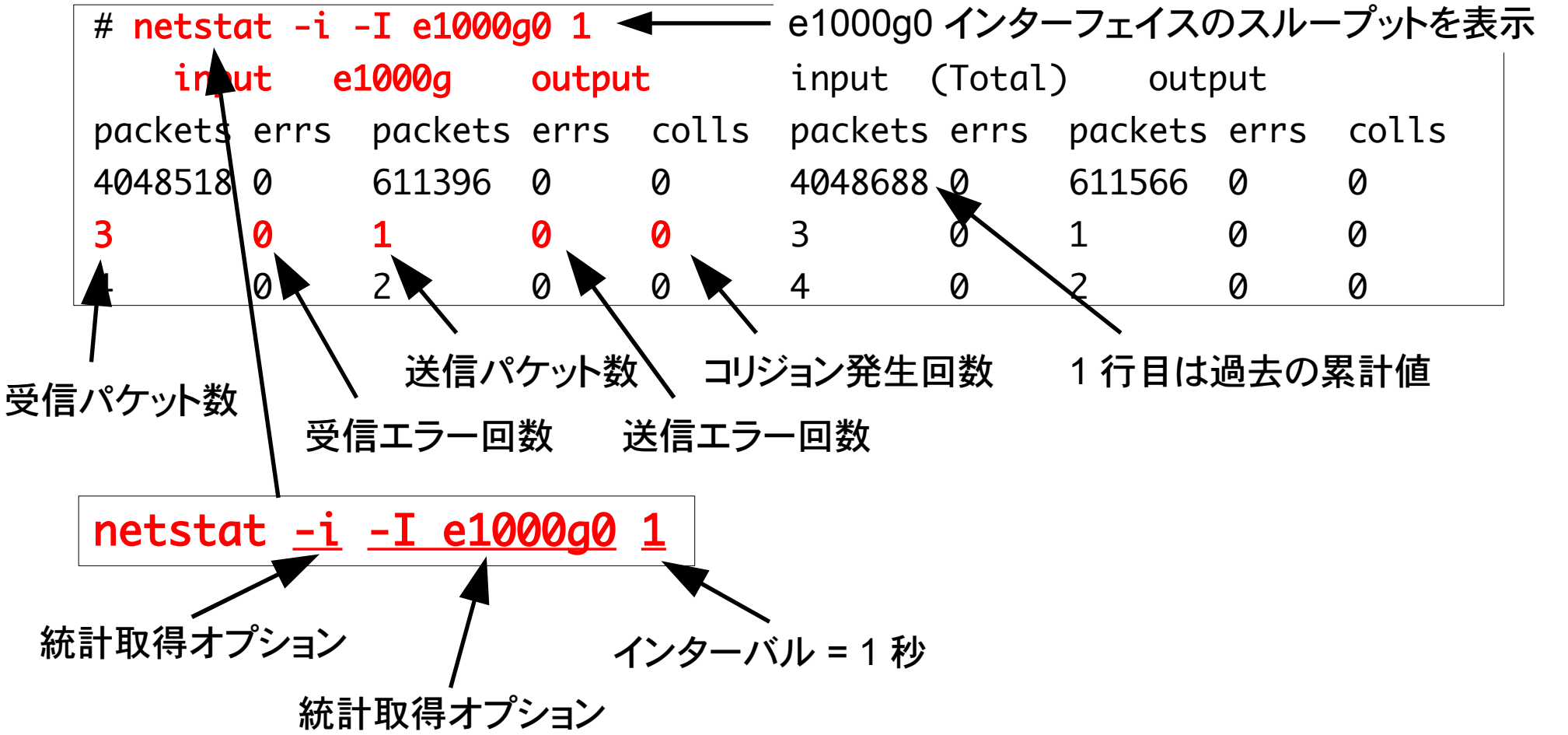
- "netstat -i -I <NIC> 1" でインターフェイス毎のスループットを表示出来ます

```
# netstat -i -I e1000g0 1
     input   e1000g     output           input  (Total)    output
packets errs  packets errs  colls  packets errs  packets errs  colls
4048518 0     611396  0     0      4048688 0     611566  0     0
3       0     1       0     0      3       0     1       0     0
4       0     2       0     0      4       0     2       0     0
```

e1000g0 インターフェイスのスループットを表示

# Tips #3 NIC のスループットの表示

- "netstat -i -I <NIC> 1" でインターフェイス毎のスループットを表示出来ます



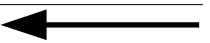
# 注意!

- netstat コマンドは非常によく使われていますが、スループットの測定で最も重要な項目は**送受信バイト数**です
- netstat コマンドでは送受信バイト数を調べる事が**出来ません**
- 一方、次にご説明する dladm コマンドや kstat コマンドを使用すると、送受信バイト数を調べる事が出来ます
- よって、極力 dladm や kstat を使用される事をお勧めします

# Tips #3 NIC のスループットの表示

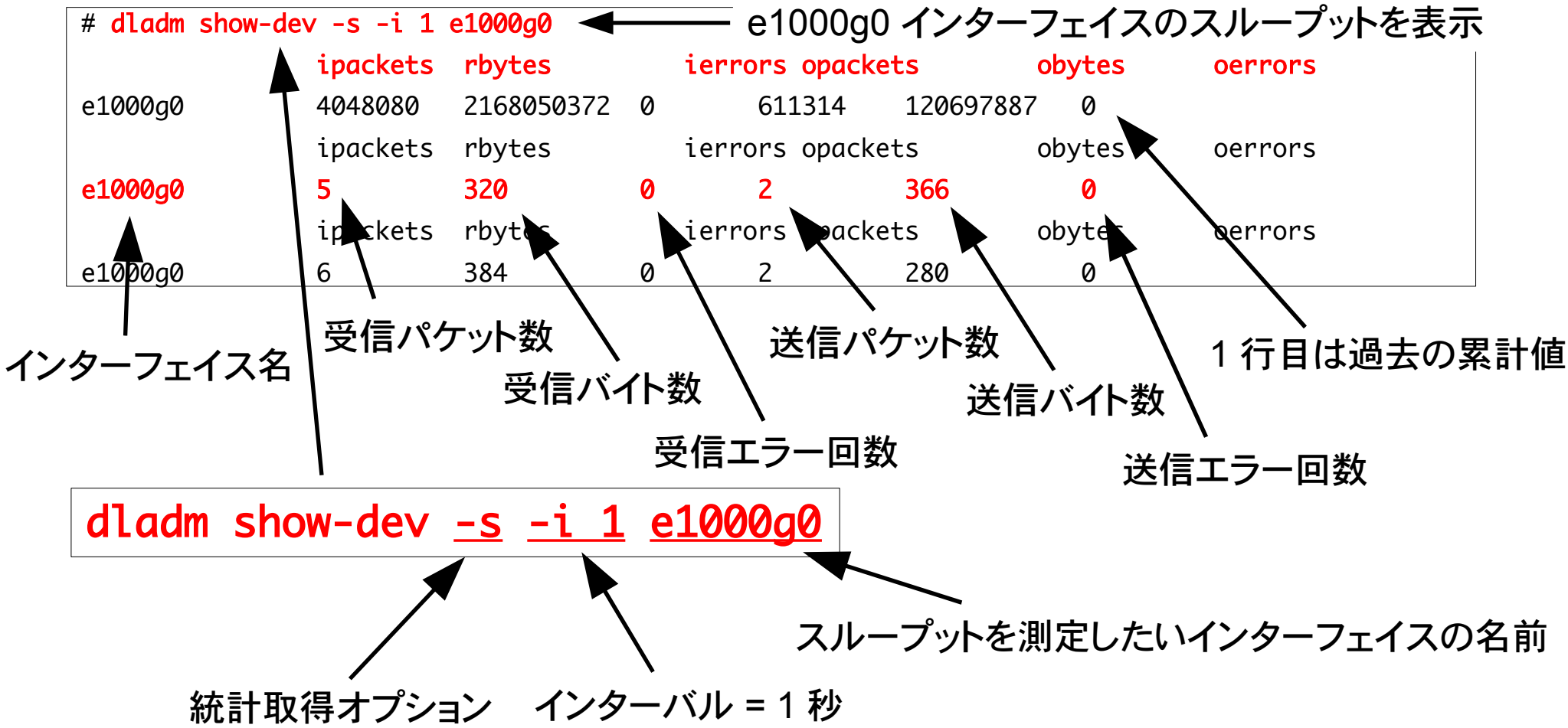
- "dladm show-dev -s -i 1 <NIC>" でインターフェイス毎のスループットを表示出来ます

```
# dladm show-dev -s -i 1 e1000g0
                ipackets  rbytes         ierrors opackets        obytes      oerrors
e1000g0         4048080   2168050372  0       611314    120697887   0
                ipackets  rbytes         ierrors opackets        obytes      oerrors
e1000g0         5         320         0       2         366         0
                ipackets  rbytes         ierrors opackets        obytes      oerrors
e1000g0         6         384         0       2         280         0
```



# Tips #3 NIC のスループットの表示

- "dladm show-dev -s -i 1 <NIC>" でインターフェイス毎のスループットを表示出来ます



# Tips #3 NIC のスループットの表示

- kstat コマンド でインターフェイス毎のスループットを表示出来ます

```
# kstat -p 'e1000g:0:e1000g0:*bytes' 1
e1000g:0:e1000g0:obytes 120717403
e1000g:0:e1000g0:rbytes 2168116829

e1000g:0:e1000g0:obytes 120717532
e1000g:0:e1000g0:rbytes 2168117213

e1000g:0:e1000g0:obytes 120717789
e1000g:0:e1000g0:rbytes 2168117597


# kstat -p 'e1000g:0:e1000g0:*packets' 1
e1000g:0:e1000g0:ipackets       4050305
e1000g:0:e1000g0:opackets       611558

e1000g:0:e1000g0:ipackets       4050309
e1000g:0:e1000g0:opackets       611559

e1000g:0:e1000g0:ipackets       4050315
e1000g:0:e1000g0:opackets       611561
```

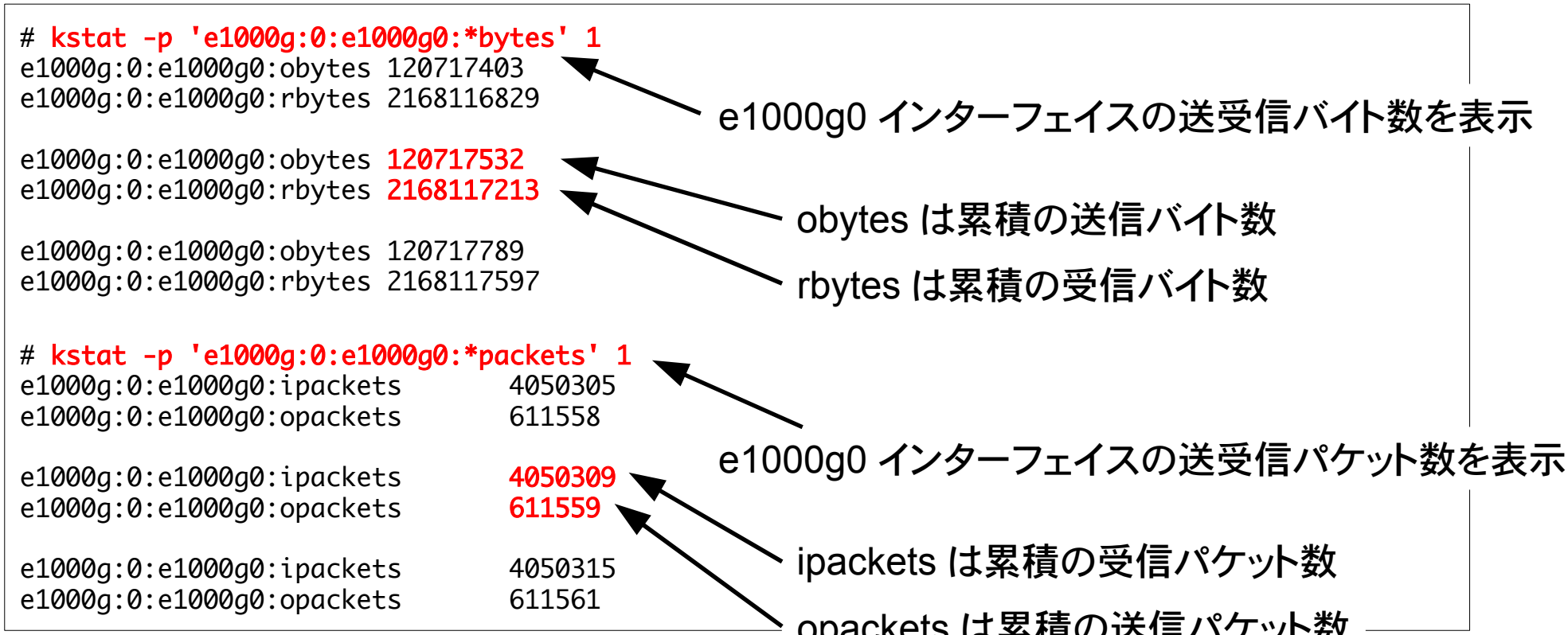
e1000g0 インターフェイスの送受信バイト数を表示
インターバルは 1 秒

e1000g0 インターフェイスの送受信パケット数を表示
インターバルは 1 秒

# Tips #3 NIC のスループットの表示

- kstat コマンド でインターフェイス毎のスループットを表示出来ます

```
# kstat -p 'e1000g:0:e1000g0:*bytes' 1
e1000g:0:e1000g0:obytes 120717403
e1000g:0:e1000g0:rbytes 2168116829

e1000g:0:e1000g0:obytes 120717532
e1000g:0:e1000g0:rbytes 2168117213

e1000g:0:e1000g0:obytes 120717789
e1000g:0:e1000g0:rbytes 2168117597

# kstat -p 'e1000g:0:e1000g0:*packets' 1
e1000g:0:e1000g0:ipackets       4050305
e1000g:0:e1000g0:opackets       611558

e1000g:0:e1000g0:ipackets       4050309
e1000g:0:e1000g0:opackets       611559

e1000g:0:e1000g0:ipackets       4050315
e1000g:0:e1000g0:opackets       611561
```



# Tips #3

## ネットワークインターフェイス毎のスループットを調べる方法

```
# netstat -i -I <NIC> 1
# dladm show-dev -s -i 1 <NIC>
# kstat -p 'e1000g:0:e1000g0:*bytes' 1
# kstat -p 'e1000g:0:e1000g0:*packets' 1
```

~~ お薦めは dladm ですが、
慣れると kstat も非常に強力です ~~

ORACLE

# Tips #4

## ネットワークインターフェイスの CPU 使用率を見る方法

~~ NIC の処理で CPU を使い尽くして
いないかを確認します ~~

ORACLE

# Tips #4 NIC の CPU 使用率を見る

- "intrstat 1" を実行するとデバイスが発生させたインタラプトの処理に使用された CPU 時間を調べる事が出来ます

```
# intrstat 1        <-- デバイス毎の CPU 使用率を出力
...
      device |      cpu0 %tim      cpu1 %tim      cpu2 %tim      cpu3 %tim
-------------+------------------------------------------------------------
      ahci#0 |         0  0.0         0  0.0         0  0.0         0  0.0
    e1000g#0 |         0  0.0         0  0.0         0  0.0         0  0.0
    e1000g#2 |         0  0.0         0  0.0         0  0.0         0  0.0
      ehci#0 |         0  0.0         0  0.0         0  0.0         0  0.0
      uhci#0 |         0  0.0         0  0.0         0  0.0         0  0.0
      uhci#1 |         0  0.0         0  0.0         0  0.0         0  0.0
      uhci#2 |         0  0.0         0  0.0         0  0.0         0  0.0
      uhci#3 |         0  0.0         0  0.0         0  0.0         0  0.0

      device |      cpu4 %tim      cpu5 %tim      cpu6 %tim      cpu7 %tim
-------------+------------------------------------------------------------
      ahci#0 |         0  0.0         2  0.0         0  0.0         0  0.0
    e1000g#0 |         0  0.0         0  0.0      8204  5.0         0  0.0
    e1000g#2 |         0  0.0         0  0.0         1  0.0         0  0.0
      ehci#0 |        18  0.0         0  0.0         0  0.0         0  0.0
      uhci#0 |        18  0.0         0  0.0         0  0.0         0  0.0
      uhci#1 |         0  0.0         1  0.0         0  0.0         0  0.0
      uhci#2 |         0  0.0         1  0.0         0  0.0         0  0.0
      uhci#3 |         0  0.0         2  0.0         0  0.0         0  0.0
```

# Tips #4 NIC の CPU 使用率を見る

- "intrstat 1" を実行するとデバイスが発生させたインタラプトの処理に使用された CPU 時間を調べる事が出来ます

```
# intrstat 1        ←―― デバイス毎の CPU 使用率を出力
...
      device |      cpu0 %tim      cpu1 %tim      cpu2 %tim      cpu3 %tim
-------------+------------------------------------------------------------
        ahci#0 |         0  0.0         0  0.0         0  0.0         0  0.0
     e1000g#0 |         0  0.0         0  0.0         0  0.0         0  0.0
     e1000g#2 |         0  0.0         0  0.0         0  0.0         0  0.0
        ehci#0 |         0  0.0         0  0.0         0  0.0         0  0.0
        uhci#0 |         0  0.0         0
        uhci#1 |         0  0.0         0
        uhci#2 |         0  0.0         0
        uhci#3 |         0  0.0         0

      device |      cpu4 %tim      cpu5 %tim      cpu6 %tim      cpu7 %tim
-------------+------------------------------------------------------------
        ahci#0 |         0  0.0         2  0.0         0  0.0         0  0.0
     e1000g#0 |         0  0.0         0  0.0      8204  5.0         0  0.0
     e1000g#2 |         0  0.0         0  0.0         1  0.0         0  0.0
        ehci#0 |        18  0.0         0  0.0         0  0.0         0  0.0
        uhci#0 |        18  0.0         0  0.0         0  0.0         0  0.0
        uhci#1 |         0  0.0         1  0.0         0  0.0         0  0.0
        uhci#2 |         0  0.0         1  0.0         0  0.0         0  0.0
        uhci#3 |         0  0.0         2  0.0         0  0.0         0  0.0
```

e1000g0 が CPU 6 を 5% 使用

# NIC の CPU 使用率の読み方

- intrstat コマンドの出力で注意すべき点は 2 つです
  - 平均して CPU 使用率が高い
    - vmstat や mpstat でシステム全体の CPU の Idle の値が 10% 以下になっており、intrstat で NIC の CPU 使用率が平均して 30% を越えていた場合は、NIC の CPU 負荷が高いかもしれません
    - vmstat で CPU の Idle が 30% を越えていれば、通常は問題無いと思います
  - シングルスレッドが張り付いている
    - intrstat の出力において、いずれかの CPU で NIC の CPU 使用率が 80% 以上になっている場合は、ドライバのスレッドが CPU に張り付いている可能性があります
    - この場合は、カーネルパラメータを変更したり、セッション数が増える様にアプリケーションの設定を変更する等、何らかの対策が必要かもしれません
    - 勿論、この状態で性能に不満が無ければ、対策は不要です

- **注意**
  - 上記閾値は個人的な経験に基づいた数値です。計算機科学上の理論的な根拠がある訳ではありませんので、ご注意下さい。システム構成に依っても多少変わってきます。

# Tips #5

## ネットワークインターフェイスの統計情報を見る方法

~~ NIC に関する様々な統計情報にアクセスします ~~

ORACLE

# Tips #5 NIC の統計情報を見る

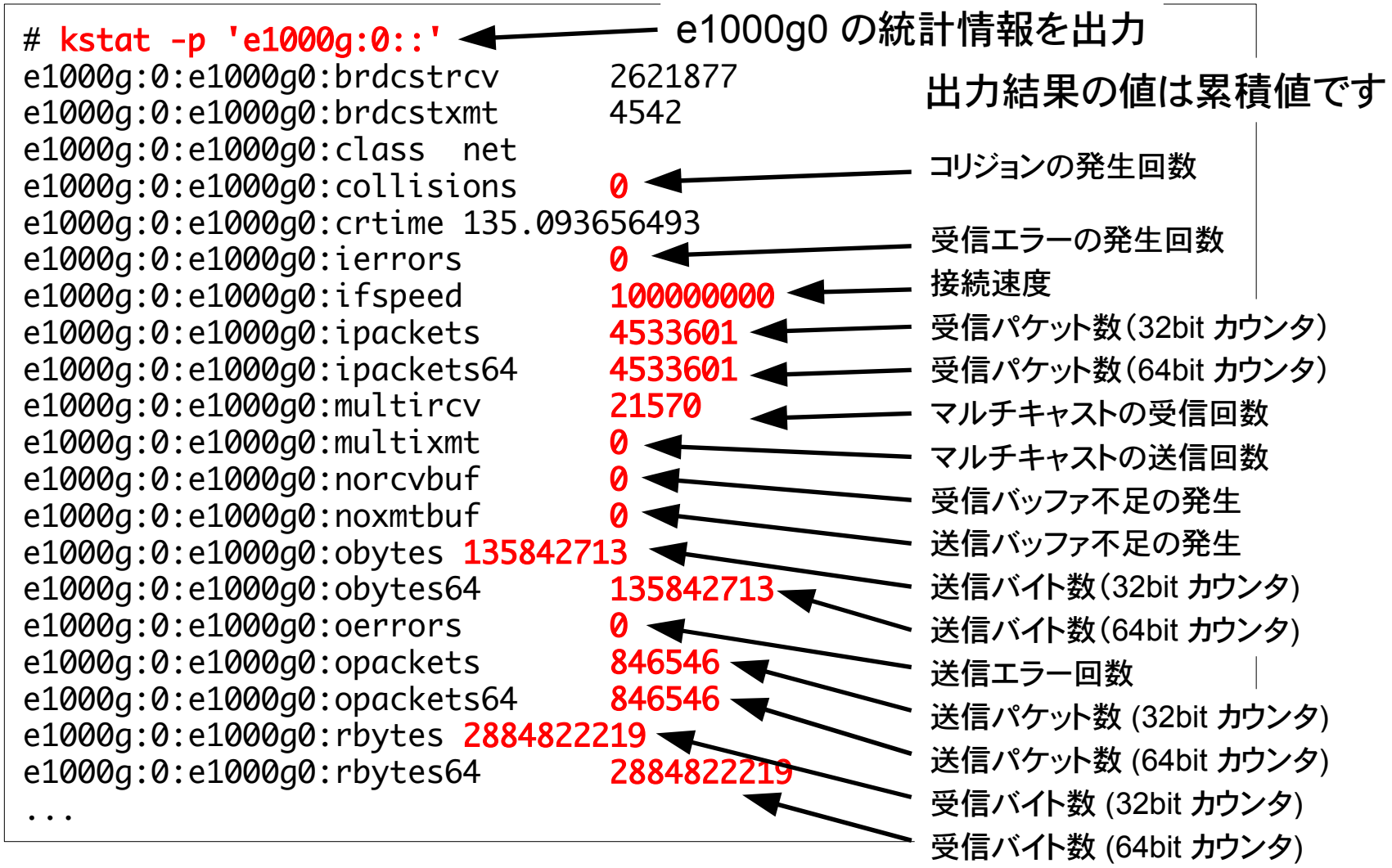
- "kstat -p '<Driver>:<Instance>::'" を実行すると

```
# kstat -p 'e1000g:0::'
e1000g:0:e1000g0:brdcstrcv      2621877
e1000g:0:e1000g0:brdcstxmt      4542
e1000g:0:e1000g0:class  net
e1000g:0:e1000g0:collisions     0
e1000g:0:e1000g0:crtime 135.093656493
e1000g:0:e1000g0:ierrors        0
e1000g:0:e1000g0:ifspeed        100000000
e1000g:0:e1000g0:ipackets       4533601
e1000g:0:e1000g0:ipackets64     4533601
e1000g:0:e1000g0:multircv       21570
e1000g:0:e1000g0:multixmt       0
e1000g:0:e1000g0:norcvbuf       0
e1000g:0:e1000g0:noxmtbuf       0
e1000g:0:e1000g0:obytes 135842713
e1000g:0:e1000g0:obytes64       135842713
e1000g:0:e1000g0:oerrors        0
e1000g:0:e1000g0:opackets       846546
e1000g:0:e1000g0:opackets64     846546
e1000g:0:e1000g0:rbytes 2884822219
e1000g:0:e1000g0:rbytes64       2884822219
...
```

e1000g0 の統計情報を出力

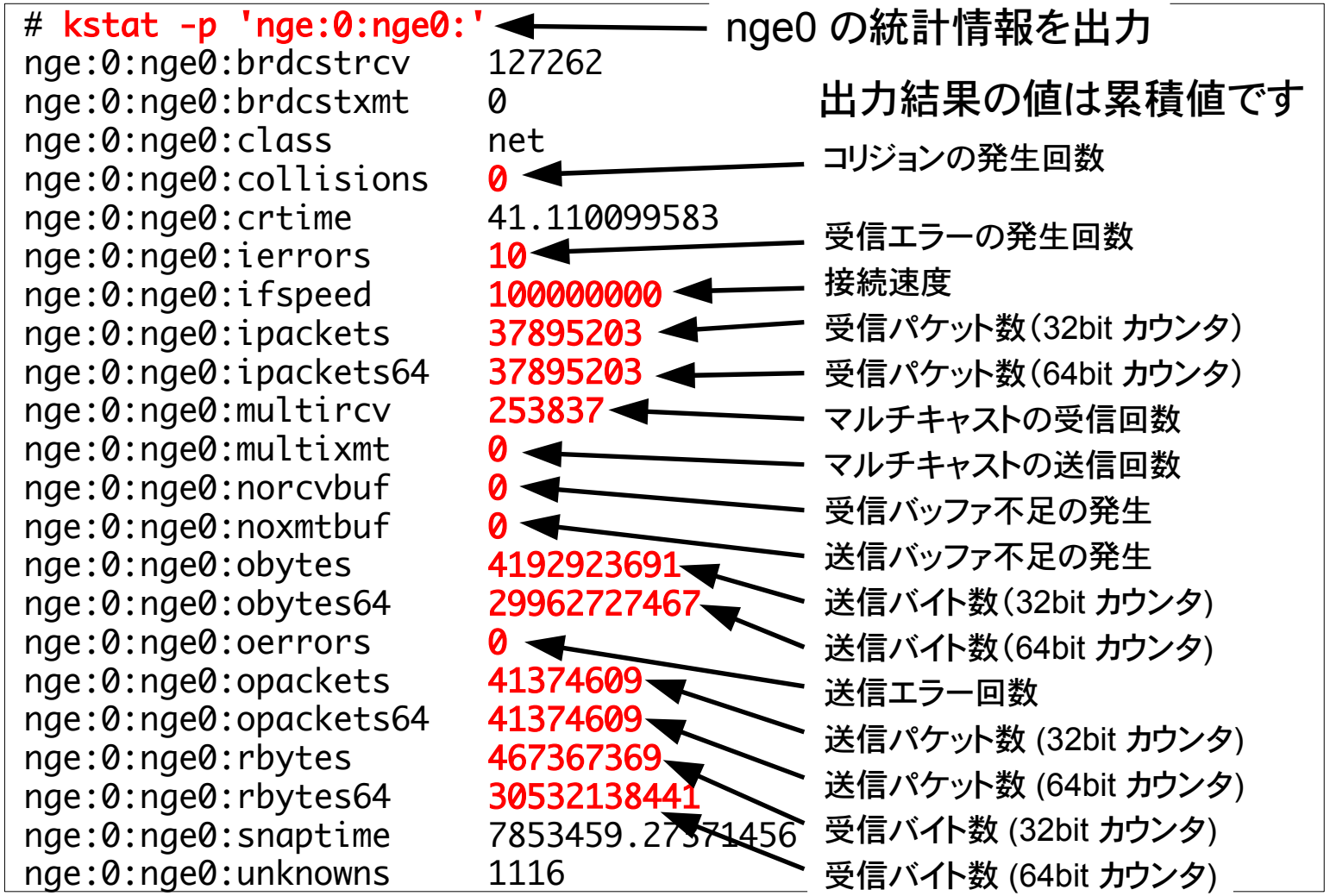
# Tips #5 NIC の統計情報を見る

- "kstat -p '<Driver>:<Instance>::'" を実行すると



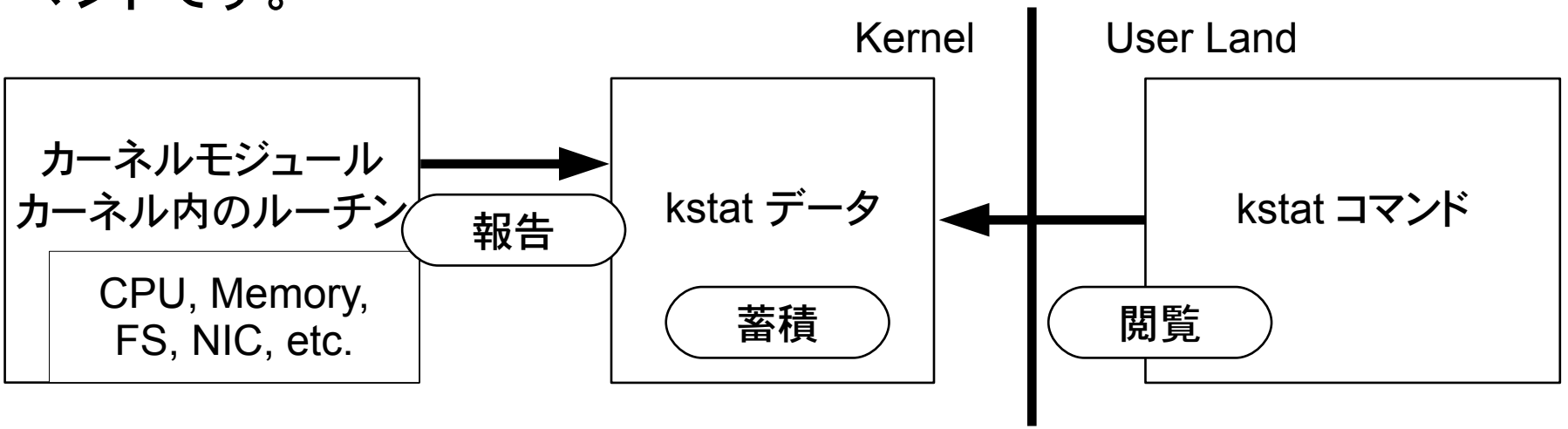
# Tips #5 NIC の統計情報を見る

- nge ドライバの場合



# kstat について

- kstat はカーネル内の様々な統計情報を一元管理する仕組みです。CPU、メモリ、ディスク I/O、ファイルシステム、スケジューラ、仮想デバイス、物理デバイス等、カーネルが管理するリソースについて、沢山の情報が kstat に常時報告されています。
- NIC に関する統計情報の多くは NIC のドライバが kstat に報告しています。報告される情報の大半は殆どの NIC に共通の物ですが、特定の NIC に固有の統計情報が見つかる場合もあります。
- kstat コマンドは、カーネルの kstat のデータにアクセスする為のコマンドです。



# Tips #5 NIC の統計情報を見る

- nxge の DMA Channel の統計情報を取得する
  - nxge は NIC とメモリの間に複数の転送経路を持っており、パケットは各経路に分散されて送受信されます。それぞれの経路に効率よく分散されているかどうかを kstat コマンドを使用して確認する事が出来ます。

```
# kstat -p 'nxge:1::*dc_packets' 1
nxge:1:RDC Channel 0 Stats:rdc_packets   17920164
nxge:1:RDC Channel 1 Stats:rdc_packets   17876325
nxge:1:RDC Channel 2 Stats:rdc_packets   18551372
nxge:1:RDC Channel 3 Stats:rdc_packets   17262833
nxge:1:RDC Channel 4 Stats:rdc_packets   17916873
nxge:1:RDC Channel 5 Stats:rdc_packets   18427140
nxge:1:RDC Channel 6 Stats:rdc_packets   18380673
nxge:1:RDC Channel 7 Stats:rdc_packets   18262768
nxge:1:TDC Channel 0 Stats:tdc_packets   25398140
nxge:1:TDC Channel 1 Stats:tdc_packets   26289370
nxge:1:TDC Channel 2 Stats:tdc_packets   25947133
nxge:1:TDC Channel 3 Stats:tdc_packets   25353460
nxge:1:TDC Channel 4 Stats:tdc_packets   27044560
nxge:1:TDC Channel 5 Stats:tdc_packets   26080875
nxge:1:TDC Channel 6 Stats:tdc_packets   25467176
nxge:1:TDC Channel 7 Stats:tdc_packets   25782669
...
```

nxge の DMA Channel の統計情報

Receive DMA Channel 情報
（各チャネルを通過した受信パケットの数）

Transmit DMA Channel 情報
（各チャネルを通過した送信パケットの数）

# Tips #5

## ネットワークインターフェイスの統計情報を見る方法

**"kstat -p '<Driver>:<Instance>::'"**

~~ 多くの NIC に共通する情報と
個別の NIC のみの情報が取得出来ます ~~

ORACLE

# Tips #6

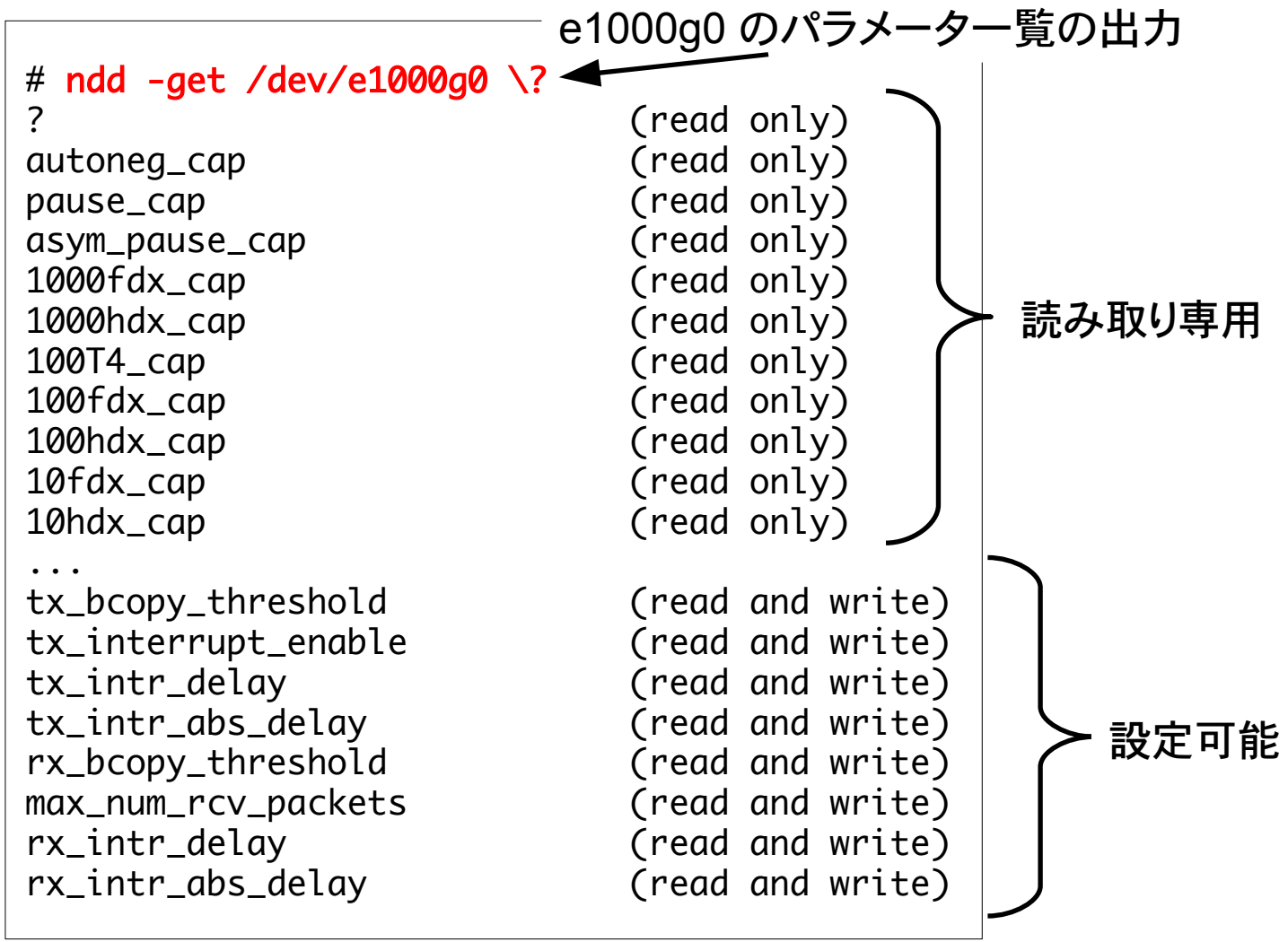
## ネットワークインターフェイスの構成情報を見る方法

~~ NIC に関する設定情報を確認します ~~

ORACLE

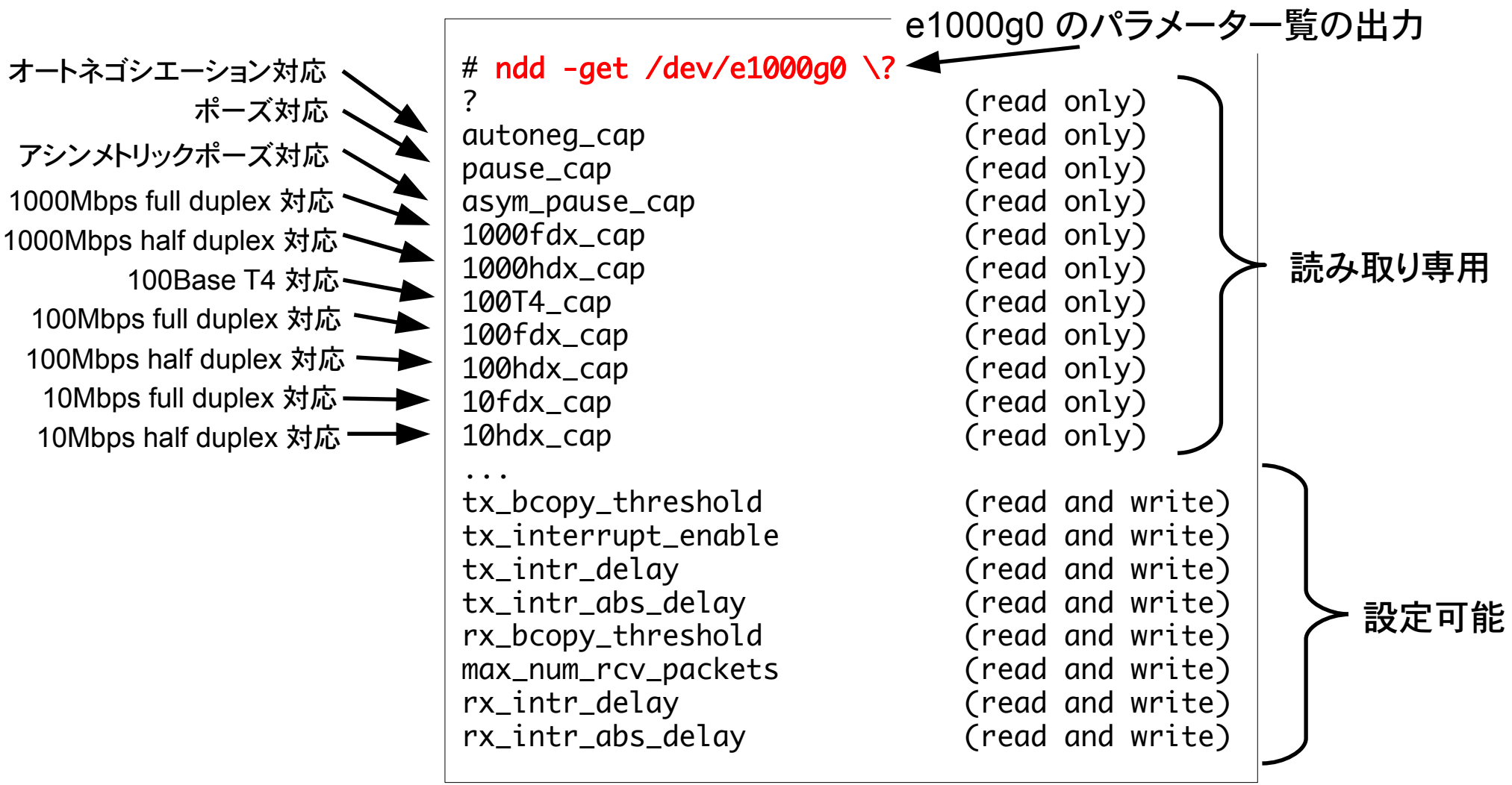
# Tips #6 NIC の構成情報を見る

- "ndd -get /dev/e1000g0 \?" を実行すると

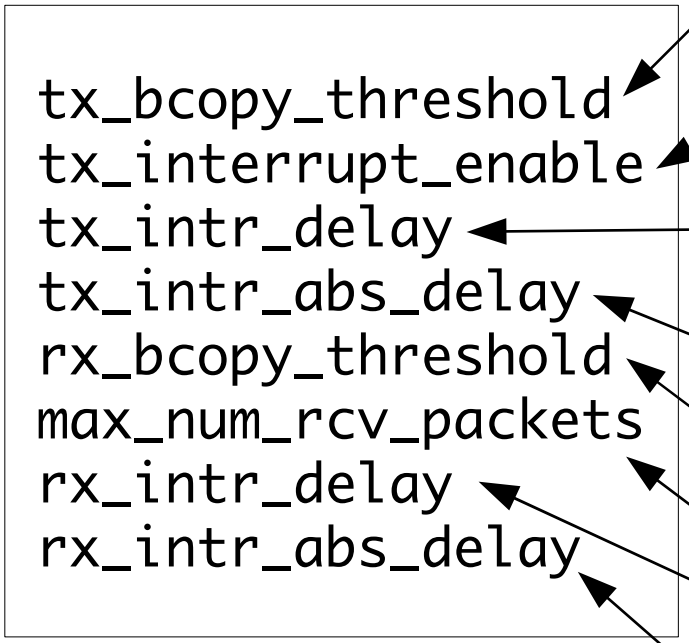


# Tips #6 NIC の構成情報を見る

- "ndd -get /dev/e1000g0 \?" を実行すると



# e1000g のパラメータの続き




tx: transmit, パケット送信時に使用される値です。
この値より大きいサイズのパケットはデータ転送に bcopy(3C) 関数が使われ、
この値より小さいサイズのパケットは DMA で転送されます。

ディスクリプタの回収にインタラプトを発生させるか
0：抑制、1：発生
この値は、NIC の E1000_IMS レジスタに格納

インタラプト発生までの遅延間隔
E1000_TIDV レジスタの値

インタラプト発生までの遅延間隔
E1000_TADV レジスタの値

インタラプトの発生を制御するパラメータ

rx: receive, パケットの受信時に使用されるパラメータです。
意味合いは tx_bcopy_threshold と一緒です。

1 回のインタラプトで処理出来るパケットの数

最後の受信データがキューに入ってから
受信インタラプトが発生するまでの遅延

最初の受信パケットがキューに入ってから
受信インタラプトが発生するまでの遅延

インタラプトの発生を制御するパラメータ

いずれもネットワークのパフォーマンスに影響するパラメータです
それぞれのパラメータの詳細は http://hub.opensolaris.org/bin/view/Project+brussels/e1000 を確認して下さい

# Tips #6 NIC の構成情報を見る

- "ndd -get /dev/e1000g0 <PARAM>" で値を参照出来ます

```
# ndd -get /dev/e1000g0 tx_bcopy_threshold
512
```

e1000g0 の tx_bcopy_threshold パラメータの値を出力

- "ndd -set /dev/e1000g0 <PARAM> <VALUE>" で値を変更出来ます (write フラグのあるパラメータのみ)

```
# ndd -set /dev/e1000g0 tx_bcopy_threshold 1024
# ndd -get /dev/e1000g0 tx_bcopy_threshold
1024
```

e1000g0 の tx_bcopy_threshold パラメータの値を変更

ndd コマンドは OS の再起動を必要とせずにカーネルモジュールのパラメータを変更するコマンドです

# Tips #6

## ネットワークインターフェイスの構成情報を見る方法

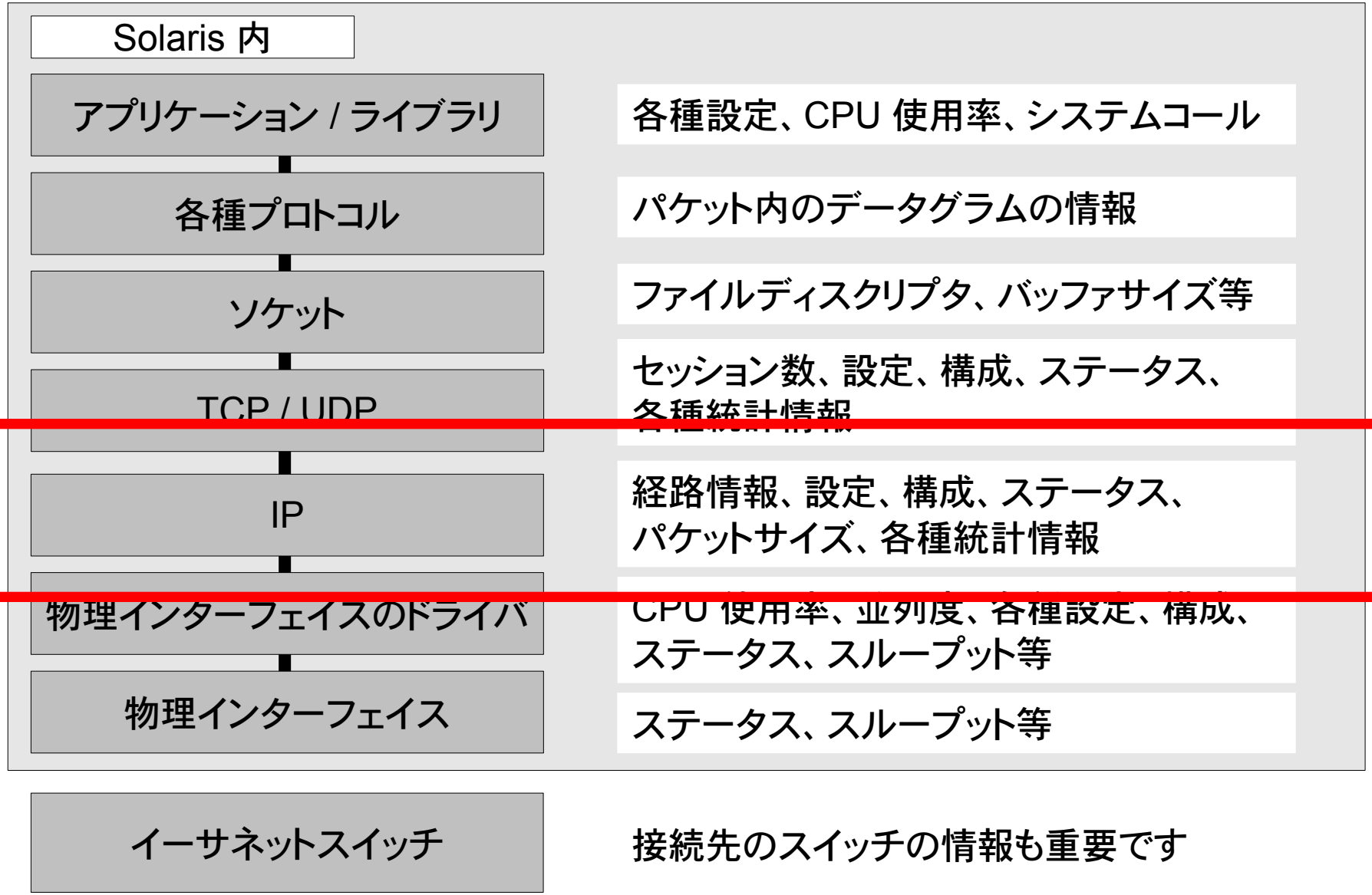
“ndd -get <NIC Device> \?”

“ndd -get <NIC Device> <Parameter>”

~~ 多くの NIC に共通する情報と
個別の NIC のみの情報が取得出来ます ~~

ORACLE

# 更に一段上がり、ここからは IP のレイヤーです



# Tips #7

## 別のホストに到達可能かを調べる

~~ ルーティング情報を調査します ~~

ORACLE

# Tips #7 別のホストに到達可能かを調べる

- "ping <HOST>" を実行すると、指定したホストまで到達可能か確認する事が出来ます。

```
# ping 10.16.62.7
10.16.62.7 is alive
```

10.16.62.7 に到達可能であるかを確認

到達可能

- "ping -s <HOST>" を実行すると、定期的に ping を打つ事が出来ます。

```
# ping -s 10.16.62.7
PING 10.16.62.7: 56 data bytes
64 bytes from 10.16.62.7: icmp_seq=0. time=0.872 ms
64 bytes from 10.16.62.7: icmp_seq=1. time=0.280 ms
64 bytes from 10.16.62.7: icmp_seq=2. time=0.214 ms
^C
----10.16.62.7 PING Statistics----
3 packets transmitted, 3 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev = 0.214/0.455/0.872/0.36
```

10.16.62.7 に到達可能であるかを確認

到達可能

到達可能

到達可能

# Tips #7 別のホストに到達可能かを調べる

- ブロードキャストアドレス宛に ping を打つと同じセグメントに居るマシンを調べる事が出来ます。

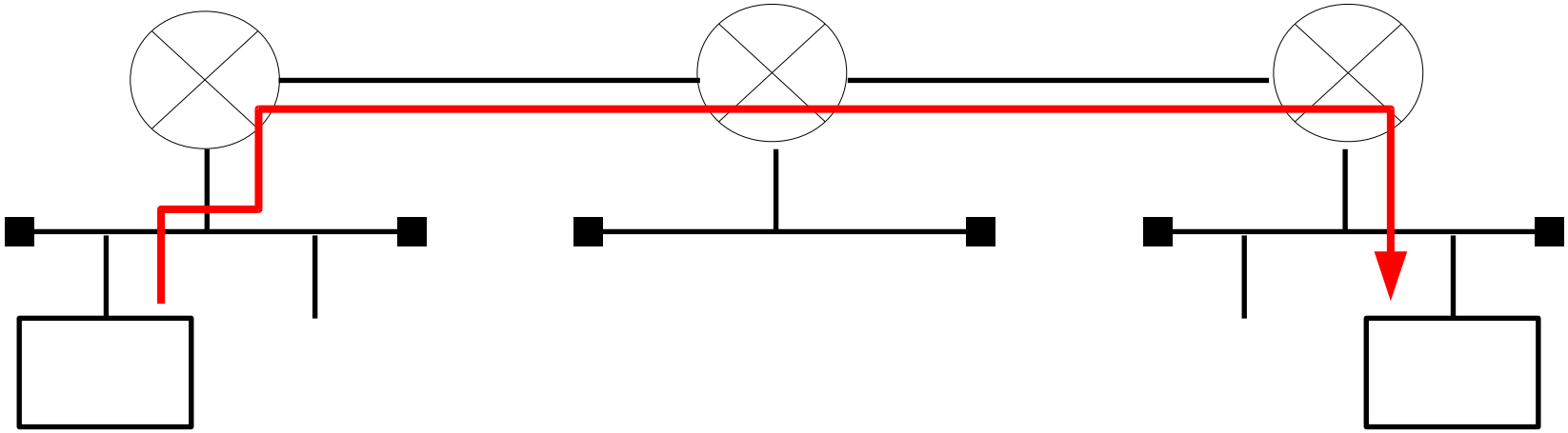
```
# ping -s 10.16.255.255          ←―― ブロードキャストアドレスに ping
PING 10.16.255.255: 56 data bytes
64 bytes from sanuki05.jp.iforce.net (10.16.67.5): icmp_seq=0.
time=0.0850 ms
64 bytes from 10.16.11.15: icmp_seq=0. time=0.325 ms  ←―― 到達可能
64 bytes from 10.16.65.51: icmp_seq=0. time=0.485 ms  ←―― 到達可能
64 bytes from 10.16.11.13: icmp_seq=0. time=0.581 ms  ←―― 到達可能
64 bytes from 10.16.62.65: icmp_seq=0. time=0.666 ms  ←―― 到達可能
...
```

注）/dev/ip の ip_respond_to_echo_broadcast パラメータが 0 のマシンは応答しません

# Tips #7 別のホストに到達可能かを調べる

- "traceroute <HOSTNAME>" を実行すると、宛先ホストまでの経路を調べる事が出来ます。

```
# traceroute 10.16.0.254          <-- 10.16.0.254 までの経路を調査
traceroute to 10.16.0.254 (10.16.0.254), 30 hops max, 40 byte packets
 1  10.16.100.254 (10.16.100.254)  0.474 ms *  0.493 ms
```



# Tips #7 別のホストに到達可能かを調べる

- "netstat -r" を実行するとルーティングテーブルを表示する事が出来ます。"default" の行がデフォルトゲートウェイです。

```
# netstat -nr          ←―― ルーティングテーブルを表示

Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use      Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              10.16.0.254          UG        1       1054
10.16.0.0            10.16.67.5           U         1        202 e1000g0
224.0.0.0            10.16.67.5           U         1          0 e1000g0
127.0.0.1            127.0.0.1            UH        4         77 lo0
```

宛先ネットワーク

ゲートウェイ

フラグ:
U は経路が健在
G は宛先がゲートウェイ
H は宛先がホスト

通過パケット数

インターフェイス名

netstat -nrv を実行すると、より詳細な情報を入手する事が出来ます

# Tips #7

## 別のホストに到達可能かを調べる

"ping <HOST>"
"ping -s <HOST>"
"ping -s <BROADCAST>"
"traceroute <HOSTNAME>"
"netstat -r"

~~ 通信経路の決定は IP レイヤーの
最も大切な仕事の一つです ~~

ORACLE

# Tips #8

## IP の統計情報を観察する

ORACLE

# Tips #8 IP の統計情報を見る

- "netstat -sP ip 1" を実行すると IP の統計情報を表示する事が出来ます。

```
% netstat -sP ip 1          <-- IP の統計情報を出力
...
IPv4    ipForwarding        =     2     ipDefaultTTL        =   255
        ipInReceives        =     2     ipInHdrErrors       =     0
        ipInAddrErrors      =     0     ipInCksumErrs       =     0
        ipForwDatagrams     =     0     ipForwProhibits     =     0
        ipInUnknownProtos   =     0     ipInDiscards        =     0
        ipInDelivers        =     2     ipOutRequests       =     2
        ipOutDiscards       =     0     ipOutNoRoutes       =     0
        ipReasmTimeout      =     0     ipReasmReqds        =     0
        ipReasmOKs          =     0     ipReasmFails        =     0
        ipReasmDuplicates   =     0     ipReasmPartDups     =     0
        ipFragOKs           =     0     ipFragFails         =     0
        ipFragCreates       =     0     ipRoutingDiscards   =     0
        tcpInErrs           =     0     udpNoPorts          =     0
        udpInCksumErrs      =     0     udpInOverflows      =     0
        rawipInOverflows    =     0     ipsecInSucceeded    =     0
        ipsecInFailed       =     0     ipInIPv6            =     0
        ipOutIPv6           =     0     ipOutSwitchIPv6     =     0
```

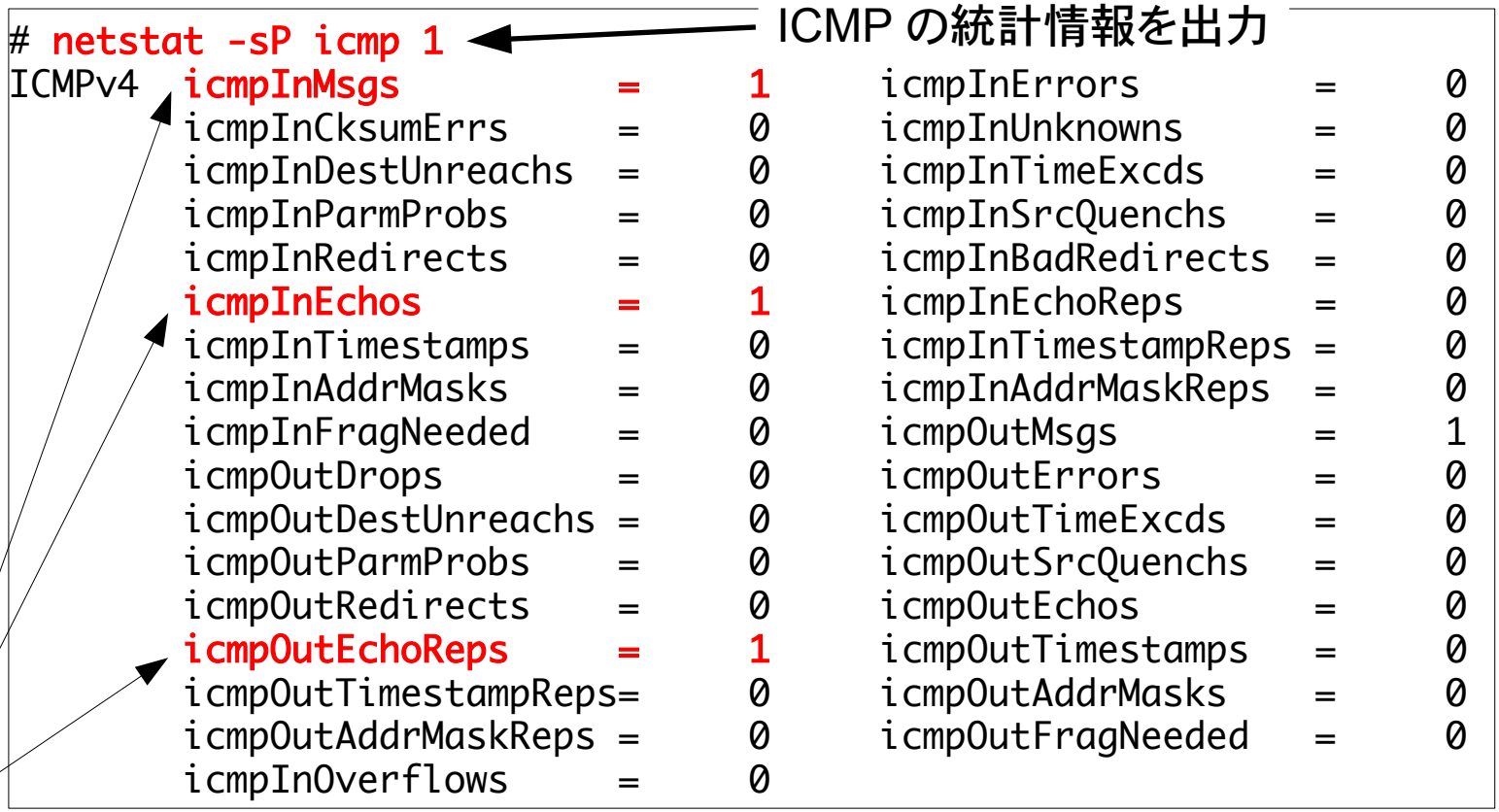
# Tips #8 IP の統計情報を見る

- "kstat -p 'ip:::' 1" を実行すると、カーネル内に保存されている IP に関する統計>情報を出力する事が出来ます。これが "netstat -sP ip 1" の出力の元データです。

```
% kstat -p 'ip:::' 1
...
ip:0:ip:inDelivers      4629963
ip:0:ip:inDiscards      0
ip:0:ip:inErrs  0
ip:0:ip:inHdrErrors     0
ip:0:ip:inIPv6  0
ip:0:ip:inReceives      4862389
ip:0:ip:inUnknownProtos 21643
...
```

# Tips #8 IP の統計情報を見る

- 同様に "netstat -sP icmp 1" を実行すると ICMP の統計情報も見る事が出来ます




```
# netstat -sP icmp 1
ICMPv4  icmpInMsgs         =     1     icmpInErrors         =     0
        icmpInCksumErrs    =     0     icmpInUnknowns       =     0
        icmpInDestUnreachs =     0     icmpInTimeExcds      =     0
        icmpInParmProbs    =     0     icmpInSrcQuenchs     =     0
        icmpInRedirects    =     0     icmpInBadRedirects   =     0
        icmpInEchos        =     1     icmpInEchoReps       =     0
        icmpInTimestamps   =     0     icmpInTimestampReps  =     0
        icmpInAddrMasks    =     0     icmpInAddrMaskReps   =     0
        icmpInFragNeeded   =     0     icmpOutMsgs          =     1
        icmpOutDrops       =     0     icmpOutErrors        =     0
        icmpOutDestUnreachs=     0     icmpOutTimeExcds     =     0
        icmpOutParmProbs   =     0     icmpOutSrcQuenchs    =     0
        icmpOutRedirects   =     0     icmpOutEchos         =     0
        icmpOutEchoReps    =     1     icmpOutTimestamps    =     0
        icmpOutTimestampReps=    0     icmpOutAddrMasks     =     0
        icmpOutAddrMaskReps=     0     icmpOutFragNeeded    =     0
        icmpInOverflows    =     0
```

ping を打つと数値が加算されます

# Tips #8

## IP の統計情報を観察する

"netstat -sP ip 1"
"kstat -p 'ip:::' 1"
"netstat -sP icmp 1"

ORACLE

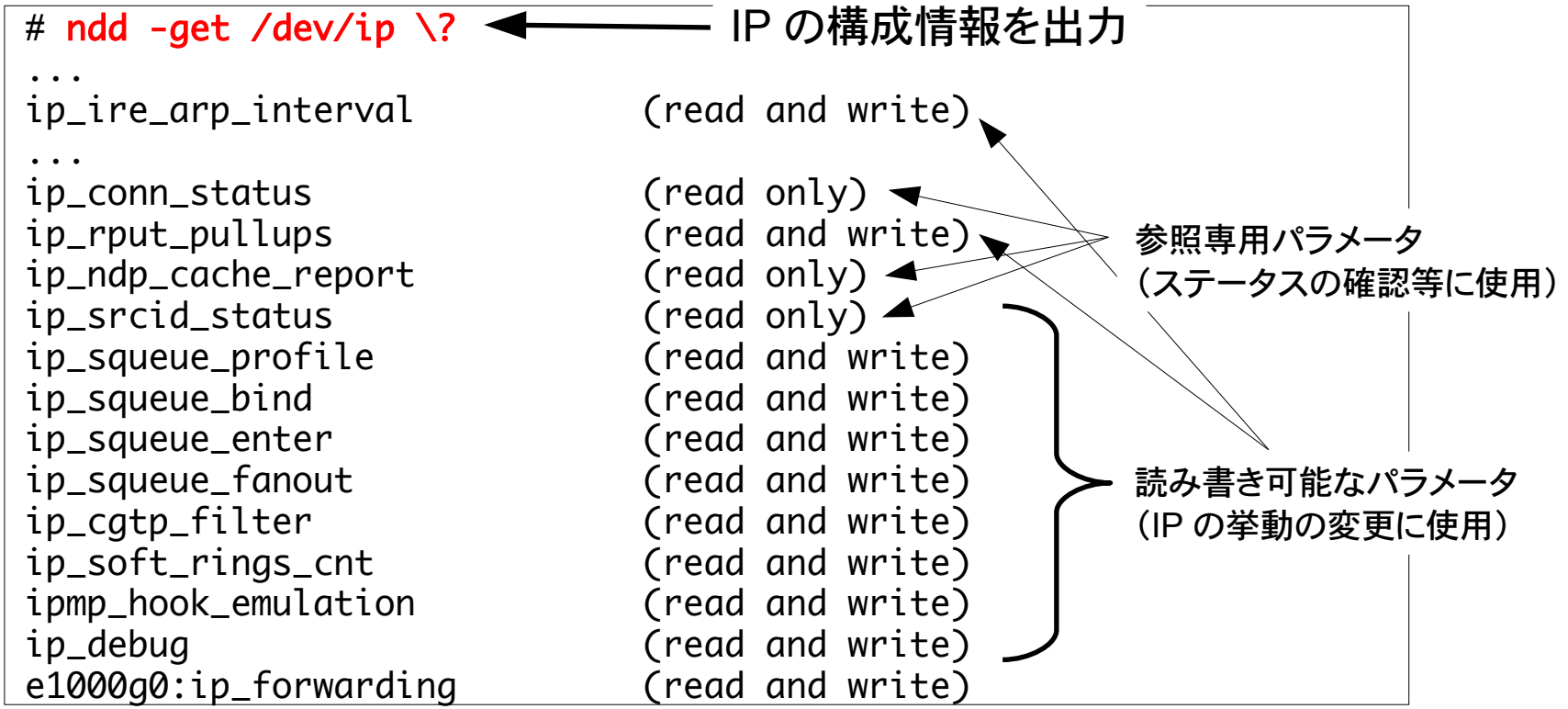
# Tips #9

## IP の構成情報を参照する

~~ Kernel の IP モジュールの設定を確認します ~~

# Tips #9 IP の構成情報を見る

- "ndd -get /dev/ip \?" を実行すると IP のパラメータを表示する事が出来ます。




それぞれのパラメータの詳細はカーネルのチューンアップ・リファレンスマニュアル
http://docs.sun.com/app/docs/doc/819-0376/chapter4-21 を確認して下さい

# Tips #9 IP の構成情報を見る

- "ndd -get /dev/ip \?" を実行すると IP のパラメータを表示する事が出来ます。

```
# ndd -get /dev/ip \?          <-- IP の統計情報を出力
...
ip_ire_arp_interval           (read and write)
...
ip_conn_status                (read only)
ip_rput_pullups               (read and write)
ip_ndp_cache_report           (read only)
ip_srcid_status               (read only)
ip_squeue_profile             (read and write)
ip_squeue_bind                (read and write)
ip_squeue_enter               (read and write)
ip_squeue_fanout              (read and write)
ip_cgtp_filter                (read and write)
ip_soft_rings_cnt             (read and write)
ipmp_hook_emulation           (read and write)
ip_debug                      (read and write)
e1000g0:ip_forwarding         (read and write)
```

IRE : internet routing entry
ルーティングテーブルのエントリーの有効期間

IP レイヤーでの接続一覧

squeue: serialization queue
インタラプトを受けた CPU 以外の CPU にコネクションを振り分けるか

IP の並列度を指定

それぞれのパラメータの詳細はカーネルのチューンアップ・リファレンスマニュアル
http://docs.sun.com/app/docs/doc/819-0376/chapter4-21 を確認して下さい

# Tips #9

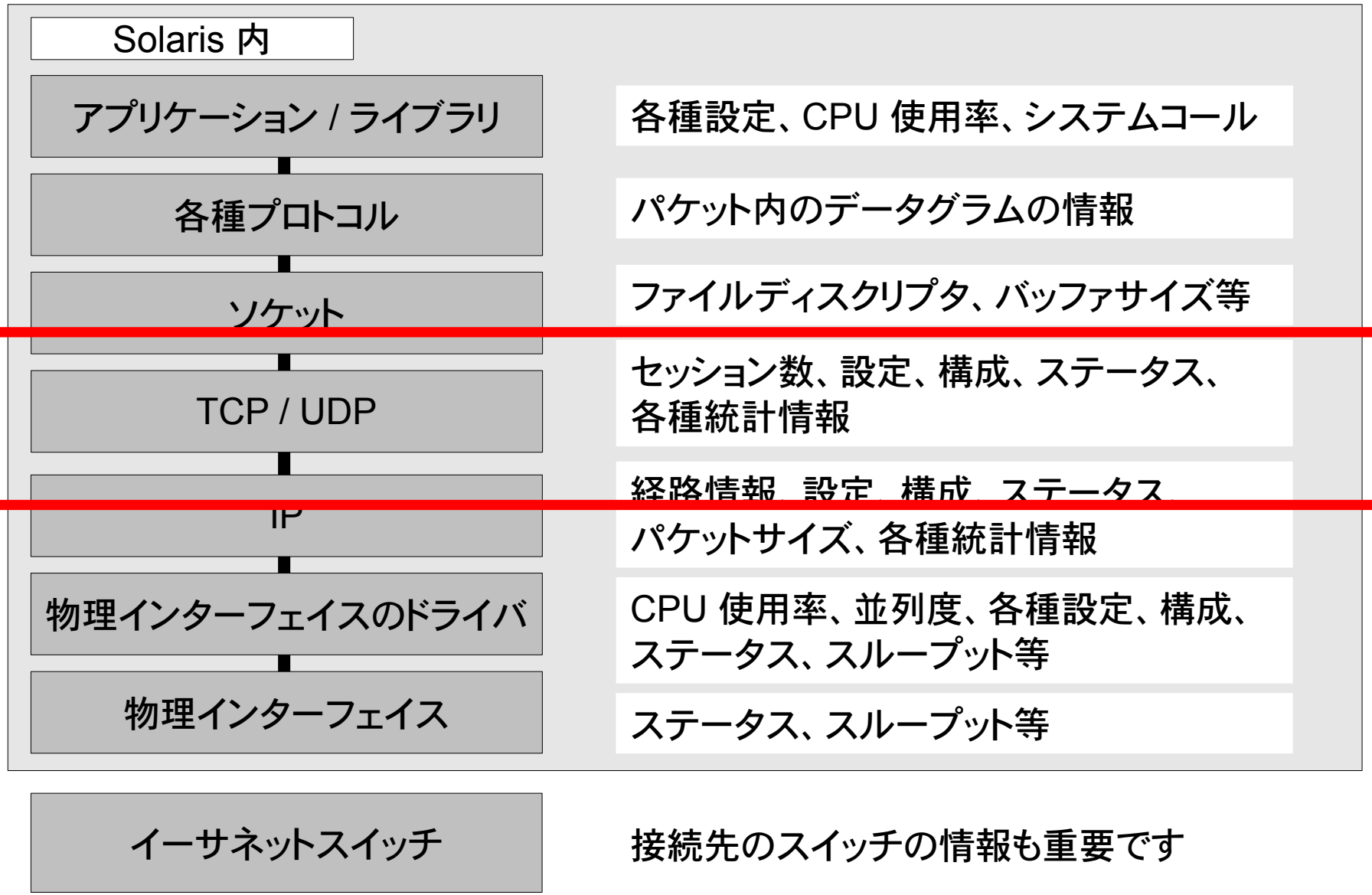
## IP の構成情報を参照する

“ndd -get /dev/ip \?”
“ndd -get /dev/ip <Parameter>”

~~ 多くの NIC に共通する情報と
個別の NIC のみの情報が取得出来ます ~~

ORACLE

# 続いて TCP/UDP のレイヤーです



# Tips #10

## ネットワークセッションの一覧を表示する方法

~~ どこから何のセッションがどのくらい張られているのかを確認します ~~

ORACLE

# Tips #10 ネットワークセッションの一覧を表示する

- "netstat -a" コマンドを使用すると現在のソケットの状態を調べる事が出来ます。State が "ESTABLISHED" になっているソケットはセッションが確立している接続です。

```
# netstat -an          <-- 開いているソケットの一覧
...
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q    State
-------------------- -------------------- ----- ------ ----- ------ -----------
      *.*                  *.*                0      0 49152      0 IDLE
10.16.67.5.22        10.16.100.16.38902   49640      0 49640      0 ESTABLISHED
      *.111                *.*                0      0 49152      0 LISTEN
      *.*                  *.*                0      0 49152      0 IDLE
      *.32771              *.*                0      0 49152      0 LISTEN
      *.32772              *.*                0      0 49152      0 LISTEN
      *.4045               *.*                0      0 49152      0 LISTEN
...
10.16.67.5.22        10.16.100.16.44023   49640      0 49640      0 ESTABLISHED
10.16.67.5.22        10.16.100.16.56928   49640      0 49640      0 ESTABLISHED
10.16.67.5.22        10.16.100.16.38539   49640     47 49640      0 ESTABLISHED
10.16.67.5.22        10.16.100.16.44189   49640      0 49640      0 ESTABLISHED
10.16.67.5.22        10.16.100.16.35588   49640      0 49640      0 ESTABLISHED
...
```

# Tips #10 ネットワークセッションの一覧を表示する

```
# netstat -an          ←―― 開いているソケットの一覧
...
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q    State
-------------------- -------------------- ----- ------ ----- ------ -----------
      *.*                  *.*                0      0 49152      0 IDLE
10.16.67.5.22        10.16.100.16.38902   49640      0 49640      0 ESTABLISHED
      *.111                *.*                0      0 49152      0 LISTEN
      *.*                  *.*                0      0 49152      0 IDLE
      *.32771              *.*                0      0 49152      0 LISTEN
      *.32772              *.*                0      0 49152      0 LISTEN
      *.4045               *.*                0      0 49152      0 LISTEN
...
10.16.67.5.22        10.16.100.16.44023   49640      0 49640      0 ESTABLISHED
10.16.67.5.22        10.16.100.16.56928   49640      0 49640      0 ESTABLISHED
10.16.67.5.22        10.16.100.16.38539   49640     47 49640      0 ESTABLISHED
10.16.67.5.22        10.16.100.16.44189   49640      0 49640      0 ESTABLISHED
10.16.67.5.22        10.16.100.16.35588   49640      0 49640      0 ESTABLISHED
...
```

自機の IP Address
及びポート番号

接続相手の IP Address
及びポート番号

S: Send, R: Receive
それぞれのウィンドウサイズ
及びシーケンス番号のギャップ

接続のステータス

netstat -anv を実行すると、より詳細な情報を入手する事が出来ます

# Tips #10 ネットワークセッションの一覧を表示する

- "netstat -aP tcp -f inet" を実行すると、IPv4 で TCP を使用しているソケットだけを抜き出す事が出来ます。

```
# netstat -aP tcp -f inet          ←――― IPv4 の情報のみを抜き出し
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q    State
-------------------- -------------------- ----- ------ ----- ------ -----------
      *.*                  *.*                0      0 49152      0 IDLE
      *.sunrpc             *.*                0      0 49152      0 LISTEN
      *.*                  *.*                0      0 49152      0 IDLE
      *.32771              *.*                0      0 49152      0 LISTEN
      *.32772              *.*                0      0 49152      0 LISTEN
      *.lockd              *.*                0      0 49152      0 LISTEN
...
```

# Tips #10 ネットワークセッションの一覧を表示する

- "netstat -a | grep ESTAB | wc -l" を実行すると、現在繋がっているネットワークセッションの本数を調べる事が出来ます。以下の様に while 分で回せば、毎秒のセッション数の増減を記録する事が出来ます。以下の例では 34 本のセッションが張られています。

```
# while :; do netstat -a | grep ESTAB | wc -l; sleep 1; done
      34
      34
      34
```

セッション数を出力

- "netstat -a | grep telnet | grep ESTAB | wc -l" を実行すると、telnet のセッションだけを抜き出す事が出来ます。同じ様に while で回せば、telnet セッションの増減を記録する事が出来ます。telnet 以外にも http や 1521 番の Oracle のセッションだけを抜き出す事も可能です。以下の例では 8 本の telnet セッションが張られています。

```
# while :; do netstat -a | grep telnet | grep ESTAB | wc -l; sleep 1;
done
       8
       8
       8
```

Telnet のセッション数を出力

# Tips #10 ネットワークセッションの一覧を表示する

- "kstat -p tcp:0:tcp:currEstab 1" を実行すると TCP のセッション数を調べる事が出来ます

```
% kstat -p tcp:0:tcp:currEstab 1
tcp:0:tcp:currEstab     4

tcp:0:tcp:currEstab     4

tcp:0:tcp:currEstab     3
```


この値は http://src.opensolaris.org/source/xref/onnv/onnv-gate/usr/src/uts/common/inet/tcp/tcp.c#25811 の tcp_kstat_update() 関数で集計されています。

# Tips #11

## TCP の統計情報を見る

~~ TCP の統計情報を観察します ~~

ORACLE

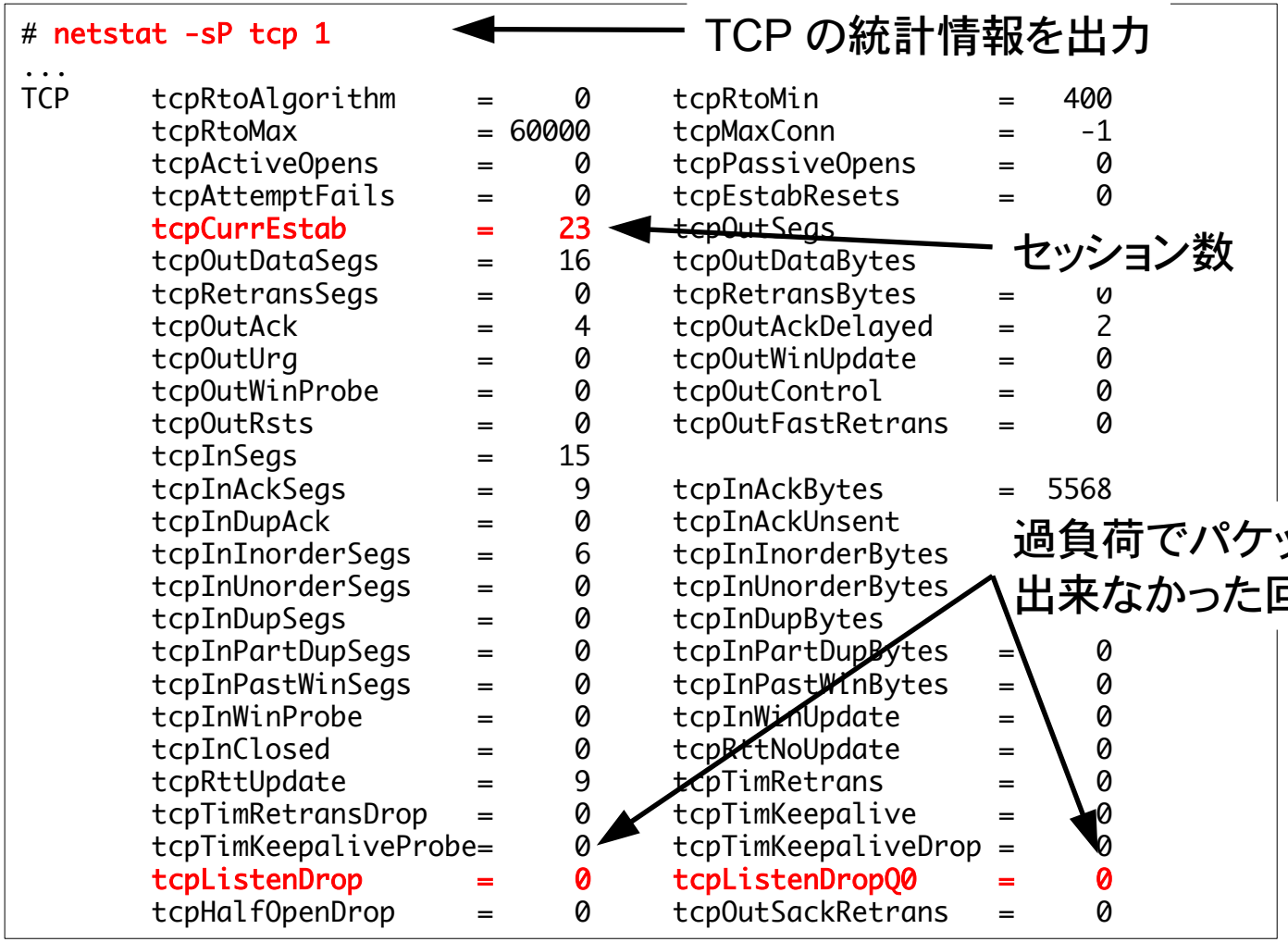
# Tips #11 TCP の統計情報を見る

- "netstat -sP tcp 1" を実行すると TCP の統計情報を表示する事が出来ます。もし tcpListenDrop が発生していたら要対応です。

```
# netstat -sP tcp 1                    <-- TCP の統計情報を出力
...
TCP     tcpRtoAlgorithm     =     0     tcpRtoMin           =   400
        tcpRtoMax           = 60000     tcpMaxConn          =    -1
        tcpActiveOpens      =     0     tcpPassiveOpens     =     0
        tcpAttemptFails     =     0     tcpEstabResets      =     0
        tcpCurrEstab        =    23     tcpOutSegs          =    20
        tcpOutDataSegs      =    16     tcpOutDataBytes     =  5568
        tcpRetransSegs      =     0     tcpRetransBytes     =     0
        tcpOutAck           =     4     tcpOutAckDelayed    =     2
        tcpOutUrg           =     0     tcpOutWinUpdate     =     0
        tcpOutWinProbe      =     0     tcpOutControl       =     0
        tcpOutRsts          =     0     tcpOutFastRetrans   =     0
        tcpInSegs           =    15
        tcpInAckSegs        =     9     tcpInAckBytes       =  5568
        tcpInDupAck         =     0     tcpInAckUnsent      =     0
        tcpInInorderSegs    =     6     tcpInInorderBytes   =  3122
        tcpInUnorderSegs    =     0     tcpInUnorderBytes   =     0
        tcpInDupSegs        =     0     tcpInDupBytes       =     0
        tcpInPartDupSegs    =     0     tcpInPartDupBytes   =     0
        tcpInPastWinSegs    =     0     tcpInPastWinBytes   =     0
        tcpInWinProbe       =     0     tcpInWinUpdate      =     0
        tcpInClosed         =     0     tcpRttNoUpdate      =     0
        tcpRttUpdate        =     9     tcpTimRetrans       =     0
        tcpTimRetransDrop   =     0     tcpTimKeepalive     =     0
        tcpTimKeepaliveProbe=     0     tcpTimKeepaliveDrop =     0
        tcpListenDrop       =     0     tcpListenDropQ0     =     0
        tcpHalfOpenDrop     =     0     tcpOutSackRetrans   =     0
```

# Tips #11 TCP の統計情報を見る

- "netstat -sP tcp 1" を実行すると TCP の統計情報を表示する事が出来ます。もし tcpListenDrop が発生していたら要対応です。




```
# netstat -sP tcp 1
...
TCP     tcpRtoAlgorithm     =     0     tcpRtoMin           =   400
        tcpRtoMax           = 60000     tcpMaxConn          =    -1
        tcpActiveOpens      =     0     tcpPassiveOpens     =     0
        tcpAttemptFails     =     0     tcpEstabResets      =     0
        tcpCurrEstab        =    23     tcpOutSegs
        tcpOutDataSegs      =    16     tcpOutDataBytes
        tcpRetransSegs      =     0     tcpRetransBytes     =     0
        tcpOutAck           =     4     tcpOutAckDelayed    =     2
        tcpOutUrg           =     0     tcpOutWinUpdate     =     0
        tcpOutWinProbe      =     0     tcpOutControl       =     0
        tcpOutRsts          =     0     tcpOutFastRetrans   =     0
        tcpInSegs           =    15
        tcpInAckSegs        =     9     tcpInAckBytes       =  5568
        tcpInDupAck         =     0     tcpInAckUnsent
        tcpInInorderSegs    =     6     tcpInInorderBytes
        tcpInUnorderSegs    =     0     tcpInUnorderBytes
        tcpInDupSegs        =     0     tcpInDupBytes
        tcpInPartDupSegs    =     0     tcpInPartDupBytes   =     0
        tcpInPastWinSegs    =     0     tcpInPastWinBytes   =     0
        tcpInWinProbe       =     0     tcpInWinUpdate      =     0
        tcpInClosed         =     0     tcpRttNoUpdate      =     0
        tcpRttUpdate        =     9     tcpTimRetrans       =     0
        tcpTimRetransDrop   =     0     tcpTimKeepalive     =     0
        tcpTimKeepaliveProbe=     0     tcpTimKeepaliveDrop =     0
        tcpListenDrop       =     0     tcpListenDropQ0     =     0
        tcpHalfOpenDrop     =     0     tcpOutSackRetrans   =     0
```

# Tips #11 TCP の統計情報を見る

- "kstat -p 'tcp:0::' 1" を実行すると TCP に関する統計情報を出力する事が出来ます。

```
% kstat -p 'tcp:0::' 1
tcp:0:tcp:activeOpens    67462
tcp:0:tcp:attemptFails   445
tcp:0:tcp:class mib2
tcp:0:tcp:connTableSize 72
tcp:0:tcp:connTableSize6         96
tcp:0:tcp:crtime         105.965466
tcp:0:tcp:currEstab      23
tcp:0:tcp:estabResets    56804
...
```

TCP の統計情報を出力

セッション数

このコマンドで取得出来るデータが "netstat -sP tcp 1" の元になっています。

# Tips #11 TCP の統計情報を見る

- DTrace Toolkit には tcpstat.d という TCP の送受信量をカウントするスクリプトが入っています。

```
# ./tcpstat.d        ←―― TCP の統計情報を出力
    TCP_out   TCP_outRe      TCP_in   TCP_inDup    TCP_inUn
          0           0           0           0           0
        122           0           0           0           0
         61           0           0           0           0
         61           0           0           0           0
     314670           0      314609           0           0
     159488           0      159427           0           0
      31213           0       31152           0           0
...
```

送信

受信

DTrace Toolkit : http://opensolaris.org/os/community/dtrace/dtracetoolkit/

# Tips #12

## TCP の構成情報を見る

ORACLE

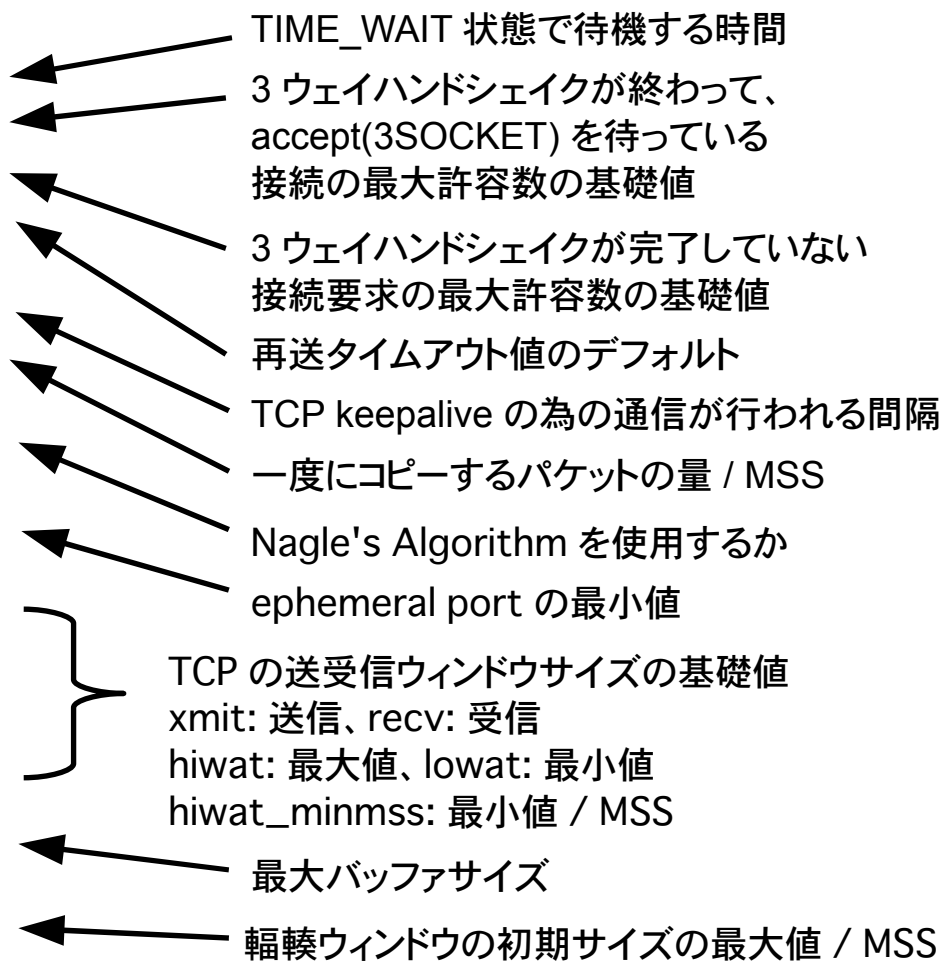
# Tips #12 TCP の構成情報を見る

- "ndd -get /dev/tcp \?" を実行すると TCP のパラメータの一部を表示する事が出来ます。

```
# ndd -get /dev/tcp \?          <-- TCP の統計情報を出力
?                               (read only)
tcp_time_wait_interval          (read and write)
tcp_conn_req_max_q              (read and write)
tcp_conn_req_max_q0             (read and write)
tcp_conn_req_min                (read and write)
tcp_conn_grace_period           (read and write)
tcp_cwnd_max                    (read and write)
tcp_debug                       (read and write)
tcp_smallest_nonpriv_port       (read and write)
...
```

# 主なパラメータ

```
...
tcp_time_wait_interval          (read and write)
tcp_conn_req_max_q              (read and write)
tcp_conn_req_max_q0             (read and write)
tcp_ip_abort_interval           (read and write)
...
tcp_keepalive_interval          (read and write)
tcp_maxpsz_multiplier           (read and write)
...
tcp_naglim_def                  (read and write)
...
tcp_smallest_anon_port          (read and write)
...
tcp_xmit_hiwat                  (read and write)
tcp_xmit_lowat                  (read and write)
tcp_recv_hiwat                  (read and write)
tcp_recv_hiwat_minmss           (read and write)
...
tcp_max_buf                     (read and write)
...
tcp_slow_start_initial          (read and write)
...
```




Solaris 9 まではこれらのパラメータを変更する事が多くありましたが、Solaris 10 以降はデフォルト値で十分な場合が多くなりました。
詳細は、TCP チューニング可能パラメータ : http://docs.sun.com/app/docs/doc/819-0376/chapter4-31 をご参照下さい。

# Tips #12 TCP の構成情報を見る

- パラメータ名を指定するとパラメータの値を取得する事が出来ます。

```
# ndd -get /dev/tcp tcp_conn_req_max_q
128
```

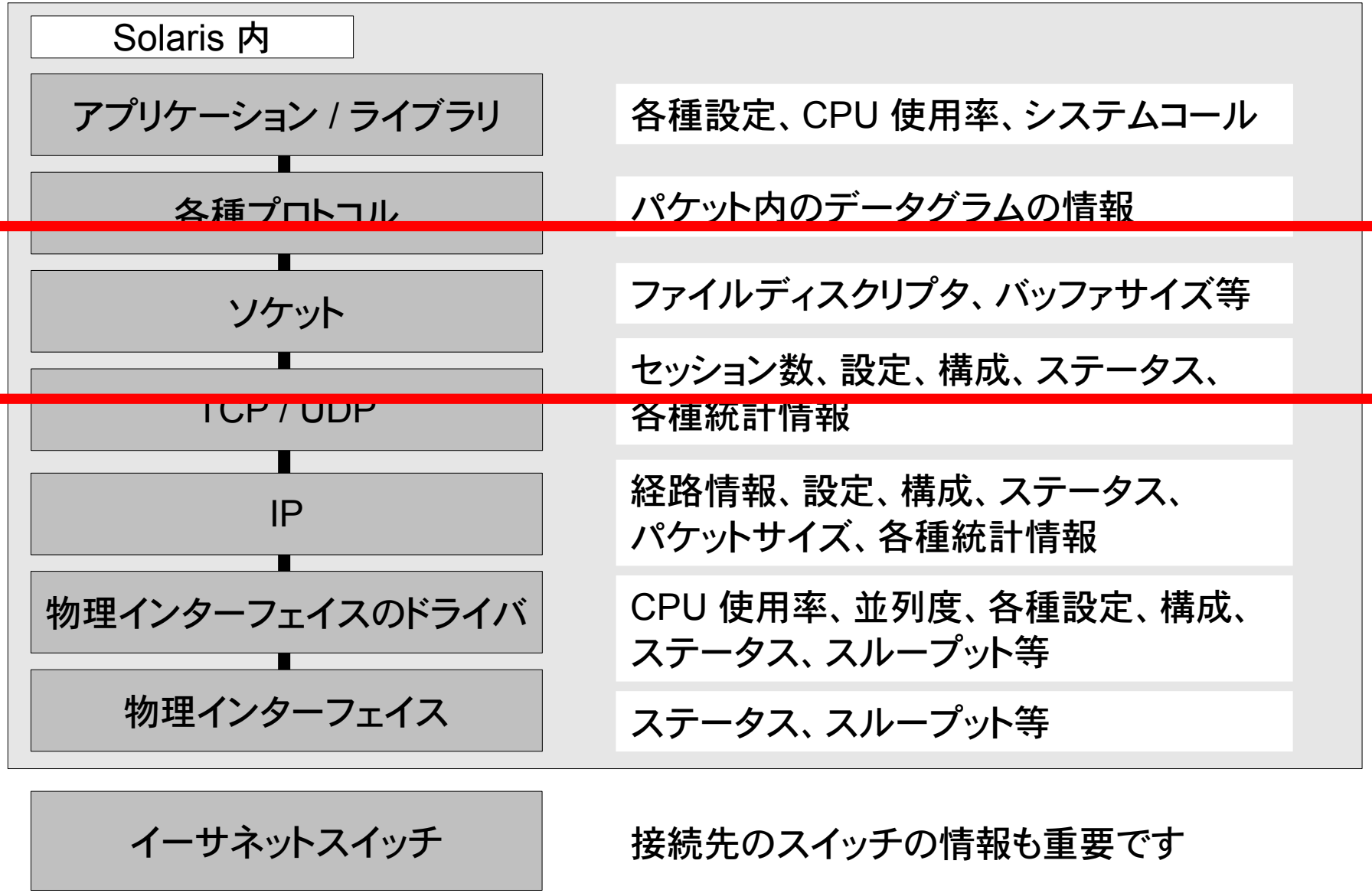
TCP の構成情報を出力

- "(read and write)" なパラメータに関しては -get の代わりに -set を指定すると値を設定する事が出来ます。

```
# ndd -set /dev/tcp tcp_conn_req_max_q 2048
# ndd -get /dev/tcp tcp_conn_req_max_q
2048
```

TCP の構成を変更

# ソケットに関する情報を調査します



# Tips #13

## プロセスが開いているソケットを調べる

~~ ソケットの情報を調べます ~~

ORACLE

# Tips #13 プロセスが開いているソケットを調べる

- "pfiles <PID>" を実行すると、プロセスが開いているソケットを調べる事が出来ます。

```
# pfiles `pgrep -x telnet`
5273:   telnet 10.16.62.7
  Current rlimit: 65536 file descriptors
   0: S_IFCHR mode:0620 dev:289,0 ino:12582920 uid:0 gid:7 rdev:24,2
      O_RDWR|O_NDELAY|O_NOCTTY|O_LARGEFILE
      /devices/pseudo/pts@0:2
 ...
   4: S_IFSOCK mode:0666 dev:296,0 ino:54580 uid:0 gid:0 size:0
      O_RDWR|O_NDELAY
        SOCK_STREAM
        SO_OOBINLINE,SO_SNDBUF(49152),SO_RCVBUF(49640),IP_NEXTHOP(232.193.0.0)
        sockname: AF_INET 10.16.67.5  port: 32793
        peername: AF_INET 10.16.62.7  port: 23
```

telnet のプロセスのソケット情報を表示

- ファイルディスクリプタ 4 番がソケットです
- ソケットオプションや接続情報を入手する事が出来ます

# Tips #13 プロセスが開いているソケットを調べる

- "pfiles <PID>" を実行すると、プロセスが開いているソケットを調べる事が出来ます。

```
# pfiles `pgrep -x telnet`
5273:   telnet 10.16.62.7
  Current rlimit: 65536 file descriptors
   0: S_IFCHR mode:0620 dev:289,0 ino:12582920 uid:0 gid:7 rdev:24,2
      O_RDWR|O_NDELAY|O_NOCTTY|O_LARGEFILE
      /devices/pseudo/pts@0:2
 ...
   4: S_IFSOCK mode:0666 dev:296,0 ino:54580 uid:0 gid:0 size:0
      O_RDWR|O_NDELAY
        SOCK_STREAM
        SO_OOBINLINE,SO_SNDBUF(49152),SO_RCVBUF(49640),IP_NEXTHOP(232.193.0.0)
        sockname: AF_INET 10.16.67.5  port: 32793
        peername: AF_INET 10.16.62.7  port: 23
```

telnet のプロセスのソケット情報を表示

ソケットのタイプ

宛先の IP Address

自分の IP Address

送信バッファサイズ

宛先のポート番号

自分側のポート番号

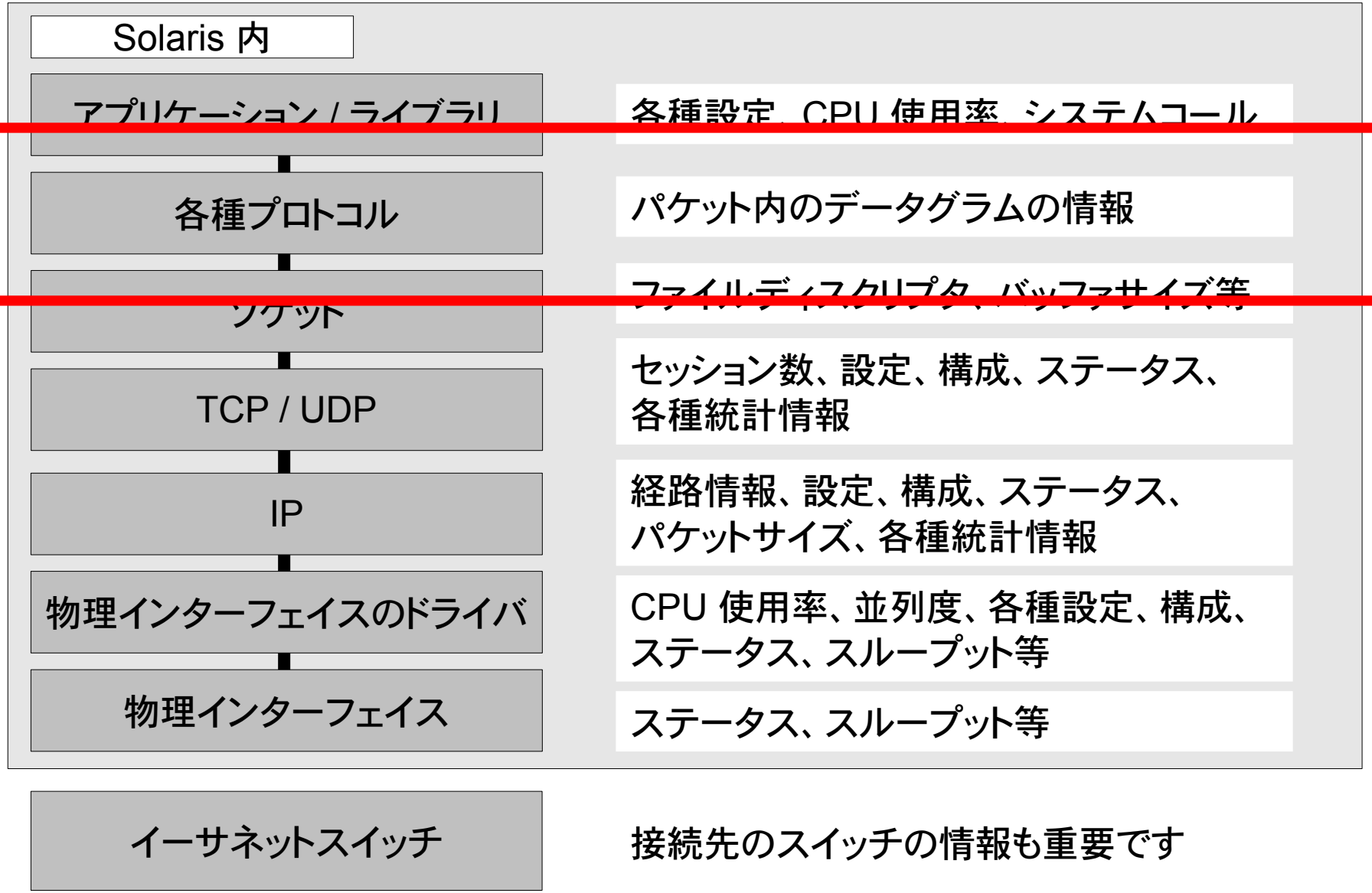
受信バッファサイズ

# Tips #13 プロセスが開いているソケットを調べる

- 以下のコマンドを実行すると、稼働中のプロセスが開いているソケットを、プロセス毎に表示させる事が出来ます。

```
 # ps -eo pid,fname | grep -v PID | while read pid fname; do echo ${fname}; pfiles ${pid} | egrep 'peername|sockname' ; done
...
java
        sockname: AF_INET 127.0.0.1  port: 6788
        sockname: AF_INET 127.0.0.1  port: 6788
        sockname: AF_INET 127.0.0.1  port: 6789
        sockname: AF_INET 127.0.0.1  port: 32787
dtlogin
        sockname: AF_INET6 ::  port: 177
        sockname: AF_INET 0.0.0.0  port: 32779
sshd
        sockname: AF_INET6 ::ffff:10.16.67.5  port: 22
        peername: AF_INET6 ::ffff:10.16.100.16  port: 38902
zsh
snmpdx
        sockname: AF_INET 0.0.0.0  port: 16161
        sockname: AF_INET 0.0.0.0  port: 32777
        sockname: AF_INET 0.0.0.0  port: 32778
...
```

# 次にパケット内の情報を調査します



# Tips #14

## どんなパケットが流れているかを調べる

~~ パケットの中身を調査します ~~

ORACLE

# Tips #14 どんなパケットが流れているかを調べる

- "snoop -d <NIC>" を実行すると、送受信したパケット毎に送信元 IP アドレス、宛先 IP アドレス、プロトコル、ポート番号等を調べる事が出来ます。

```
# snoop -r -d e1000g0
Using device /dev/e1000g0 (promiscuous mode)
10.16.100.16 -> 10.16.67.5   TELNET C port=40383
  10.16.67.5 -> 10.16.100.16 TELNET R port=40383 Using device /dev/e1
10.16.100.16 -> 10.16.67.5   TELNET C port=40383
 10.16.11.16 -> (broadcast)  ARP C Who is 10.16.0.200, 10.16.0.200 ?
  10.16.64.5 -> (broadcast)  ARP C Who is 10.16.201.102, 10.16.201.102 ?
           ? -> *            ETHER Type=9000 (Loopback), size = 60 bytes
```

e1000g0 デバイスを通るパケットを解析

- snoop -d <NIC> で <NIC> を通るパケットの中身をダンプする事が出来ます
- -r オプションは名前解決をさせない為に付けていますが、付けなくても問題ありません
- "TELNET" と表示されているパケットは telnet のパケットです
- snoop の実行はそれ自体がネットワークの負荷になりますので、必要な時にだけ使用して下さい

# Tips #14 どんなパケットが流れているかを調べる

- "snoop -d <NIC>" を実行すると、送受信したパケット毎に送信元 IP アドレス、宛先 IP アドレス、プロトコル、ポート番号等を調べる事が出来ます。

```
# snoop -r -d e1000g0
Using device /dev/e1000g0 (promiscuous mode)
10.16.100.16 -> 10.16.67.5   TELNET C port=40383
  10.16.67.5 -> 10.16.100.16 TELNET R port=40383 Using device /dev/e1
10.16.100.16 -> 10.16.67.5   TELNET C port=40383
 10.16.11.16 -> (broadcast)  ARP C Who is 10.16.0.200, 10.16.0.200 ?
  10.16.64.5 -> (broadcast)  ARP C Who is 10.16.201.102, 10.16.201.102 ?
           ? -> *            ETHER Type=9000 (Loopback), size = 60 bytes
```

e1000g0 デバイスを通るパケットを解析

送信元 IP Address　宛先 IP Address　プロトコル　追加情報：ポート番号やパケットの中身等

# Tips #14 どんなパケットが流れているかを調べる

- "snoop -d <NIC> -V" を実行するとネットワークの階層毎に詳細な情報を出力させる事が出来ます。

```
# snoop -r -d e1000g0 -V
Using device /dev/e1000g0 (promiscuous mode)
------------------------------
10.16.100.16 -> 10.16.67.5   ETHER Type=0800 (IP), size = 60 bytes
10.16.100.16 -> 10.16.67.5   IP  D=10.16.67.5 S=10.16.100.16 LEN=40, ID=44170, TOS=0x0, TTL=64
10.16.100.16 -> 10.16.67.5   TCP D=23 S=40383 Ack=2979015144 Seq=1218855275 Len=0 Win=49640
10.16.100.16 -> 10.16.67.5   TELNET C port=40383
------------------------------
  10.16.67.5 -> 10.16.100.16 ETHER Type=0800 (IP), size = 100 bytes
  10.16.67.5 -> 10.16.100.16 IP  D=10.16.100.16 S=10.16.67.5 LEN=86, ID=8412, TOS=0x0, TTL=60
  10.16.67.5 -> 10.16.100.16 TCP D=40383 S=23 Push Ack=1218855275 Seq=2979015144 Len=46 Win=49640
  10.16.67.5 -> 10.16.100.16 TELNET R port=40383 Using device /dev/e1
------------------------------
```

e1000g0 デバイスを通るパケットのサイズ

- "------------------------------" で挟まれている部分が一つのパケットです。"ETHER"、"IP"、"TCP"、"TELNET" と階層毎に一行出力されているのが分かります。

# Tips #14 どんなパケットが流れているかを調べる

- "snoop -d <NIC> -V" を実行するとネットワークの階層毎に詳細な情報を出力させる事が出来ます。

```
# snoop -r -d e1000g0 -V
Using device /dev/e1000g0 (promiscuous mode)
------------------------------
10.16.100.16 -> 10.16.67.5   ETHER Type=(
10.16.100.16 -> 10.16.67.5   IP  D=10.16                    ID=44170, TOS=0x0, TTL=64
10.16.100.16 -> 10.16.67.5   TCP D=23 S=                   8855275 Len=0 Win=49640
10.16.100.16 -> 10.16.67.5   TELNET C po
------------------------------
  10.16.67.5 -> 10.16.100.16 ETHER Type=0800 (IP), size = 100 bytes
  10.16.67.5 -> 10.16.100.16 IP  D=10.16.100.16 S=10.16.67.5 LEN=86, ID=8412, TOS=0x0, TTL=60
  10.16.67.5 -> 10.16.100.16 TCP D=40383 S=23 Push Ack=1218855275 Seq=2979015144 Len=46 Win=49640
  10.16.67.5 -> 10.16.100.16 TELNET R port=40383 Using device /dev/e1
------------------------------
```

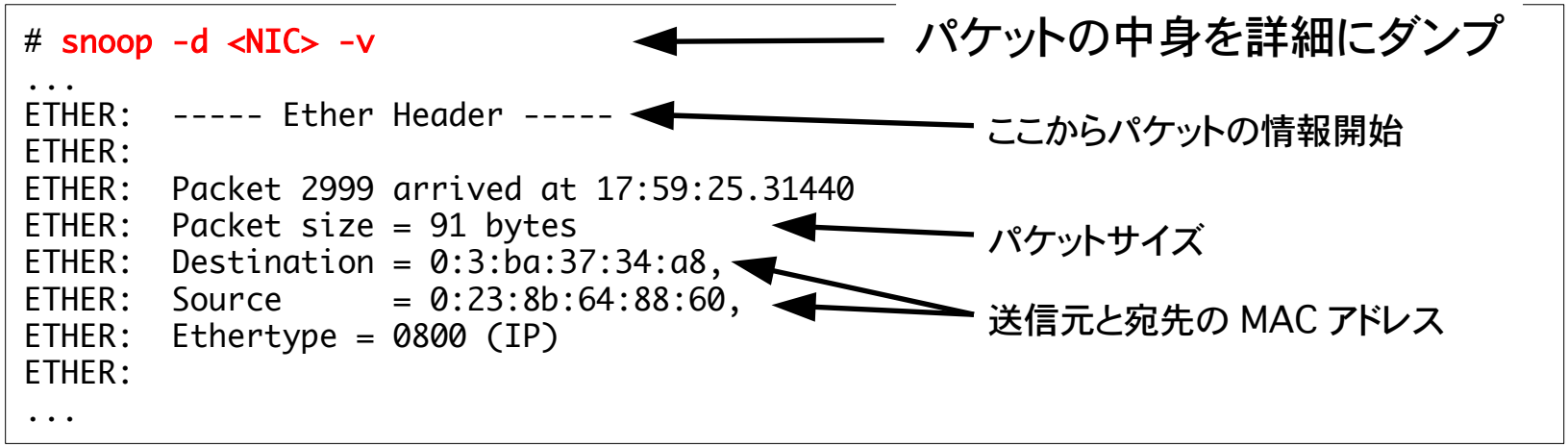
e1000g0 デバイスを通るパケットのサイズ

これが一つのパケット

送信元アドレス　宛先アドレス　プロトコル　詳細情報

# Tips #14 どんなパケットが流れているかを調べる

- "snoop -d <NIC> -v" を実行すると更に詳しくパケットの中身を調べる事が出来ます。



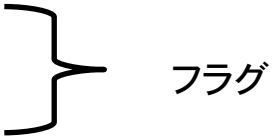

- 各パケットについて、プロトコル用のヘッダと実データが分かりやすく整形されて出力されます。
- 上記は Ethernet ヘッダの情報ですが、これに続いて、次ページの IP、TCP、ペイロードの情報が出力されていきます。

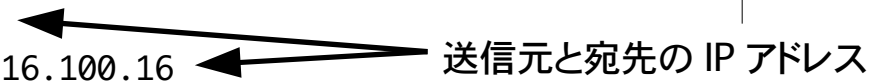
# 続いて IP ヘッダ

```
...
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:         .... ..0. = not ECN capable transport
IP:         .... ...0 = no ECN congestion experienced
IP:   Total length = 77 bytes
IP:   Identification = 18244
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 60 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 0000
IP:   Source address = 10.16.67.5, 10.16.67.5
IP:   Destination address = 10.16.100.16, 10.16.100.16
IP:   No options
IP:
...
```







# TCP ヘッダとペイロード

```
...
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 23
TCP:  Destination port = 40383
TCP:  Sequence number = 2990317620
TCP:  Acknowledgement number = 1218855394
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x18
TCP:        0... .... = No ECN congestion window reduced
TCP:        .0.. .... = No ECN echo
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 1... = Push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 49640
TCP:  Checksum = 0xbb74
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
TELNET:  ----- TELNET:   -----
TELNET:
TELNET:  "IP:   ----- IP Header -----\r\nIP:   \r\n"
TELNET:
...
```

ここから TCP の情報開始

送信元と宛先のポート番号

ここから実データの情報開始

Telnet の通信内容

# Tips #14 どんなパケットが流れているかを調べる

- "snoop -d <NIC> -x <OFFSET>" を実行すると、パケットの中身をダンプする事が出来ます。

```
# snoop -r -d e1000g0 -x 54 port 23
Using device /dev/e1000g0 (promiscuous mode)
10.16.100.16 -> 10.16.67.5   TELNET C port=40386 i

             0: 6900 0000 0000                                  i.....

  10.16.67.5 -> 10.16.100.16 TELNET R port=40386 i

             0: 69                                              i

10.16.100.16 -> 10.16.67.5   TELNET C port=40386

             0: 0000 0000 0000                                  ......

10.16.100.16 -> 10.16.67.5   TELNET C port=40386 d

             0: 6400 0000 0000                                  d.....

  10.16.67.5 -> 10.16.100.16 TELNET R port=40386 d

             0: 64                                              d
```

オフセットは 54bytes、更に port 23 番のみ表示

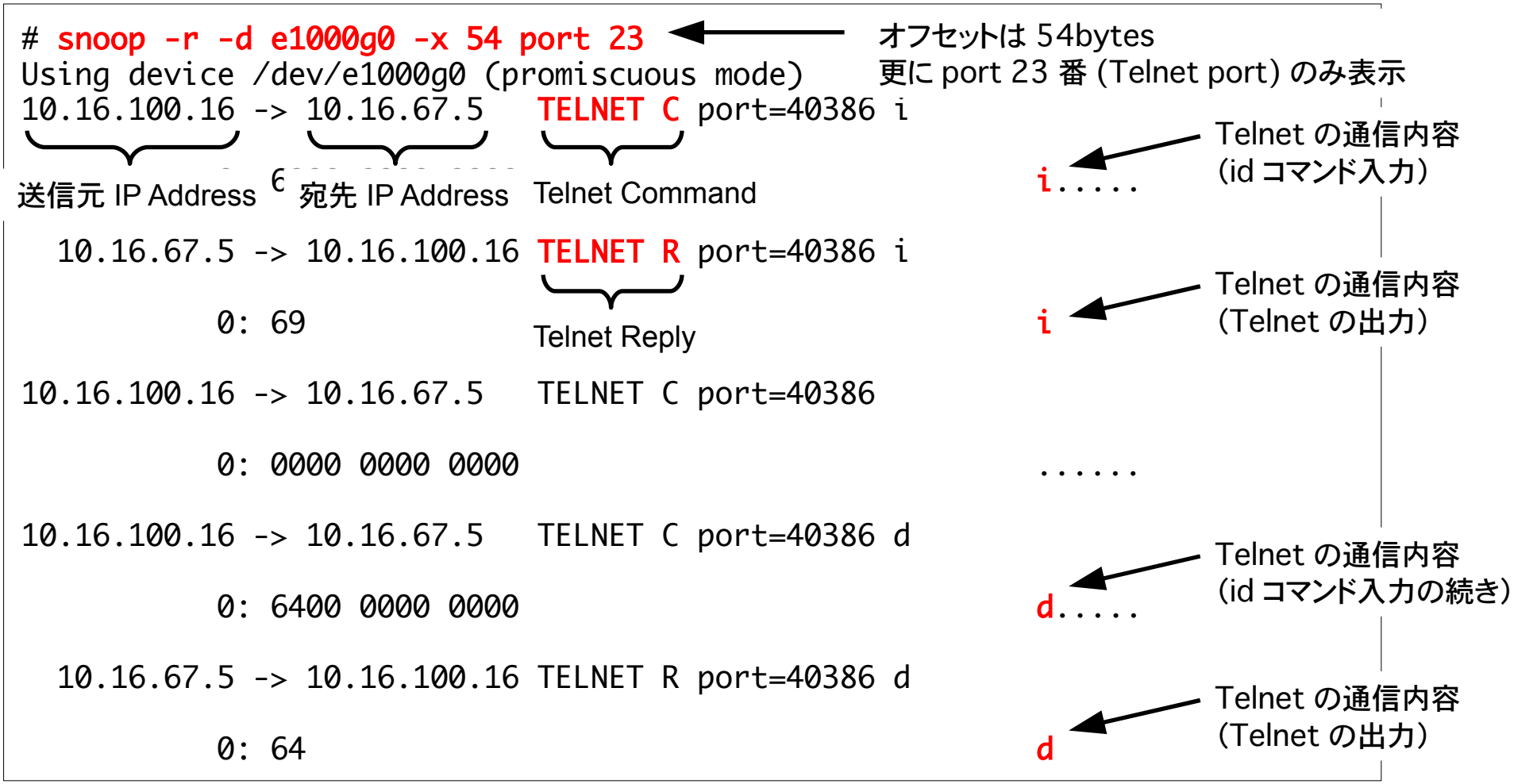
# Tips #14 どんなパケットが流れているかを調べる

- 『telnet 越しに id コマンドを実行する』というシチュエーションを例に取って snoop の出力を見てみます
- id コマンドの出力例:

```
# id
uid=0(root) gid=0(root)
```

# Tips #14 どんなパケットが流れているかを調べる

- 以下は telnet 越しに id コマンドを実行し、それを snoop で観察した例です



# snoop の出力の続き

```
10.16.100.16 -> 10.16.67.5   TELNET C port=40386

           0: 0000 0000 0000                                    ......

10.16.100.16 -> 10.16.67.5   TELNET C port=40386

           0: 0d00 0000 0000                                    ......

  10.16.67.5 -> 10.16.100.16 TELNET R port=40386

           0: 0d0a                                              ..

10.16.100.16 -> 10.16.67.5   TELNET C port=40386

           0: 0000 0000 0000                                    ......

  10.16.67.5 -> 10.16.100.16 TELNET R port=40386 uid=0(root) gid=0(ro

           0: 7569 643d 3028 726f 6f74 2920 6769 643d    uid=0(root) gid=
          16: 3028 726f 6f74 290d 0a23 20                0(root)..#

10.16.100.16 -> 10.16.67.5   TELNET C port=40386

           0: 0000 0000 0000                                    ......
```

Telnet の通信内容
(id コマンドの出力)

# Tips #14 どんなパケットが流れているかを調べる

- "snoop -o <FILE> -d <NIC> を実行すると送受信したパケットを全てファイルに保存する事が出来ます。

```
# snoop -o /var/tmp/snoop01.dump -d e1000g0
Using device /dev/e1000g0 (promiscuous mode)
24 ^C
```

e1000g0 の通信内容を /var/tmp/snoop01.dump に保存

- 保存したファイルは "snoop -i <FILE>" で読み出す事が出来ます。読み出す際には、今までご紹介した snoop のオプションも合わせて使用する事が出来ます。

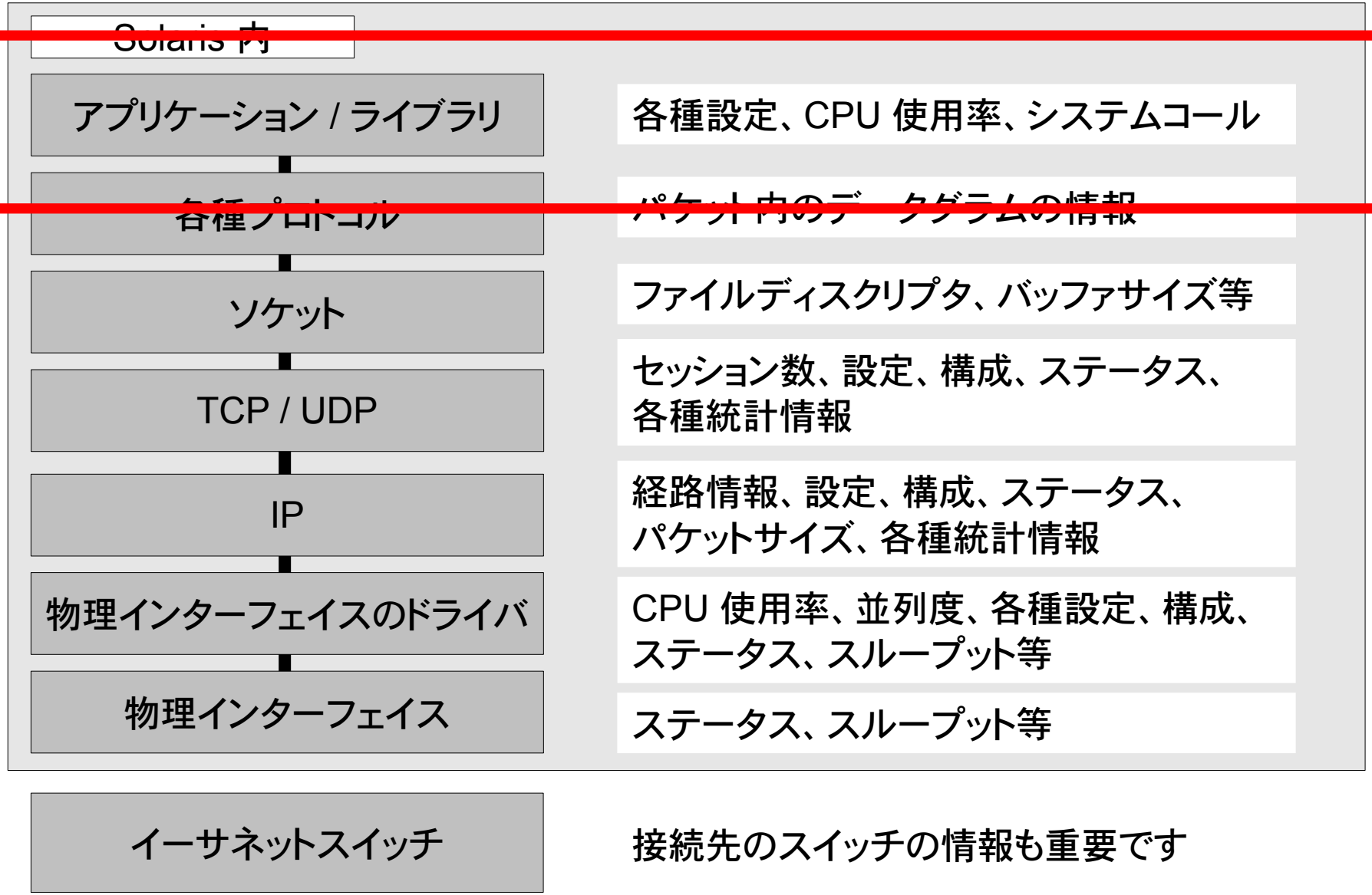
```
# snoop -r -i /var/tmp/snoop01.dump -V | head
------------------------------
  1   0.00000 10.16.100.16 -> 10.16.67.5   ETHER Type=0800 (IP), size = 60 bytes
  1   0.00000 10.16.100.16 -> 10.16.67.5   IP  D=10.16.67.5 S=10.16.100.16 LEN=40, ID=21844, TOS=0x0, TTL=64
  1   0.00000 10.16.100.16 -> 10.16.67.5   TCP D=23 S=40383 Ack=3027145344 Seq=1218855542 Len=0 Win=49640
  1   0.00000 10.16.100.16 -> 10.16.67.5   TELNET C port=40383
------------------------------
  2   0.00001   10.16.67.5 -> 10.16.100.16 ETHER Type=0800 (IP), size = 104 bytes
  2   0.00001   10.16.67.5 -> 10.16.100.16 IP  D=10.16.100.16 S=10.16.67.5 LEN=90, ID=28393, TOS=0x0, TTL=60
  2   0.00001   10.16.67.5 -> 10.16.100.16 TCP D=40383 S=23 Push Ack=1218855542 Seq=3027145344 Len=50 Win=49640
  2   0.00001   10.16.67.5 -> 10.16.100.16 TELNET R port=40383 Using device /dev
```

ファイルに保存したデータを読み出し

# 注意!

- 通信の内容を一旦ファイルに保存して後でゆっくり解析する事が出来るので非常に便利な反面、通信量に応じてファイルのサイズも大きくなりますので、ご注意下さい。

# 最後にアプリケーション側から観察します



# Tips #15

## ネットワークトラフィックに関連するシステムコールを調べる

~~ システムコールからネットワークの
状況を調査する事が出来ます ~~

ORACLE

# Tips #15 システムコールを調べる

- "truss -cp <PID>" を実行するとプロセスが発行したシステムコールとその回数を調べる事が出来ます。

```
# truss -cp `pgrep -xn sshd`        <-------- sshd が発行したシステムコールを表示
...
syscall               seconds   calls  errors
_exit                    .000       1
read                     .185    9804       1
write                    .152    9641
close                    .000       4
chmod                    .000       1
chown                    .000       1
brk                      .000       2
getuid                   .000       1
sigaction                .000       1
setcontext               .000       1
waitid                   .000       2       1
lwp_sigmask              .128   19433
pollsys                  .081    9717
stat64                   .000       1
shutdown                 .000       1
...
```

- read, write, close, pollsys, shutdown 等、ネットワークに関連するシステムコールが発行されています

# Tips #15 システムコールを調べる

- "truss -cp <PID>" を実行するとプロセスが発行したシステムコールとその回数を調べる事が出来ます。

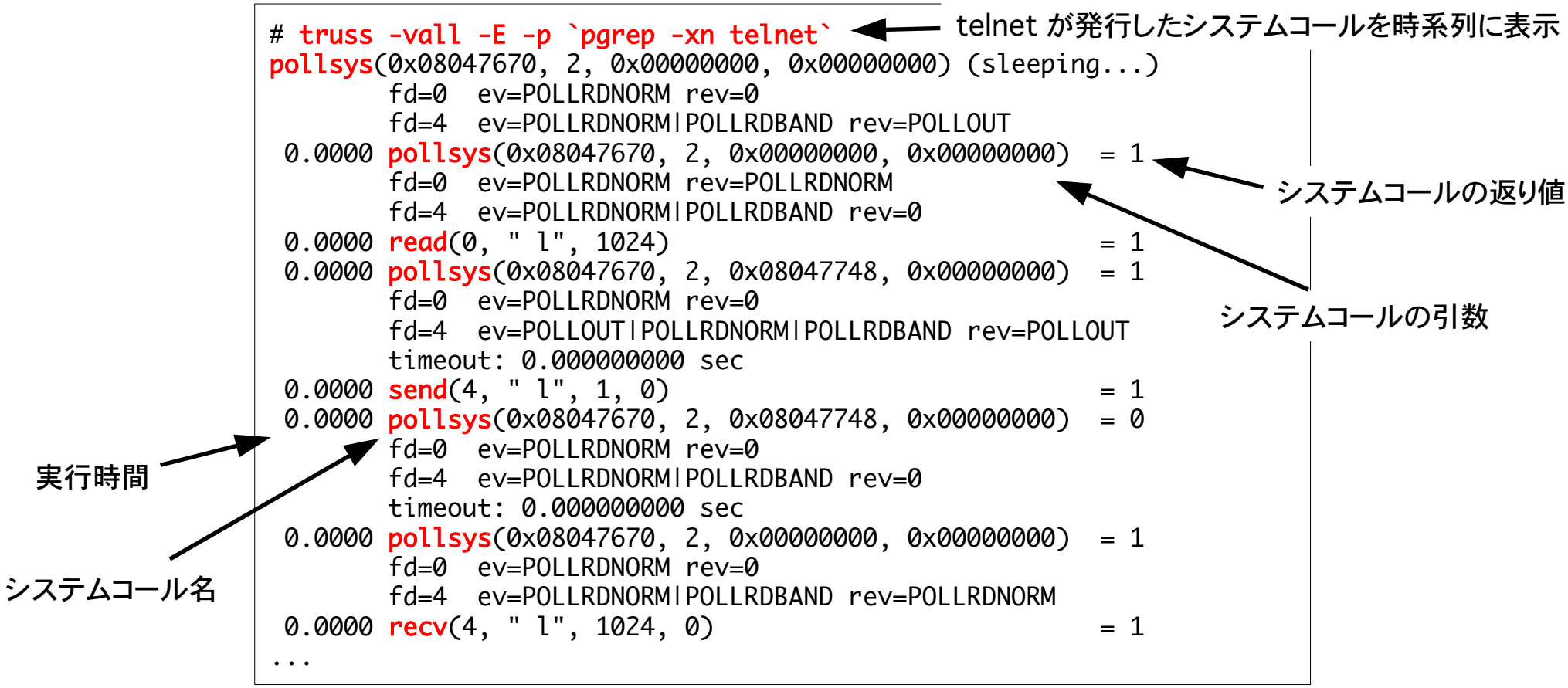
```
# truss -cp `pgrep -xn sshd`
...
syscall               seconds   calls  errors
_exit                    .000       1
read                     .185    9804       1
write                    .152    9641
close                    .000       4
chmod                    .000       1
chown                    .000       1
brk                      .000       2
getuid                   .000       1
sigaction                .000       1
setcontext               .000       1
waitid                   .000       2       1
lwp_sigmask              .128   19433
pollsys                  .081    9717
stat64                   .000       1
shutdown                 .000       1
...
```



- read, write, close, pollsys, shutdown 等、ネットワークに関連するシステムコールが発行されています
- どのシステムコールがどのくらい時間を使っているかの情報を元に、ネットワークの状況を把握する事が可能です

# Tips #15 システムコールを調べる

- "truss -vall -E -p <PID>" を実行するとプロセスが発行しているシステムコールを逐次表示する事が出来ます。




- システムコールに消費した時間、引数、返り値等も合わせて調べる事が出来るので、ネットワークの解析にも非常に有効です。

# Tips #15 システムコールを調べる

- truss の出力も snoop と同様にファイルにダンプする事が可能です。

```
# truss -o /var/tmp/truss01.dump -vall -E -p `pgrep -xn telnet`
```

truss の出力を /var/tmp/truss01.dump にダンプ

- ダンプされたファイルはテキストファイルです。

```
# head /var/tmp/truss01.dump
pollsys(0x08047670, 2, 0x00000000, 0x00000000) (sleeping...)
        fd=0  ev=POLLRDNORM rev=0
        fd=4  ev=POLLRDNORM|POLLRDBAND rev=POLLOUT
 0.0000 pollsys(0x08047670, 2, 0x00000000, 0x00000000)  = 1
        fd=0  ev=POLLRDNORM rev=POLLRDNORM
        fd=4  ev=POLLRDNORM|POLLRDBAND rev=0
 0.0000 read(0, " l", 1024)                             = 1
 0.0000 pollsys(0x08047670, 2, 0x08047748, 0x00000000)  = 1
        fd=0  ev=POLLRDNORM rev=0
        fd=4  ev=POLLOUT|POLLRDNORM|POLLRDBAND rev=POLLOUT
```

# Tips #16

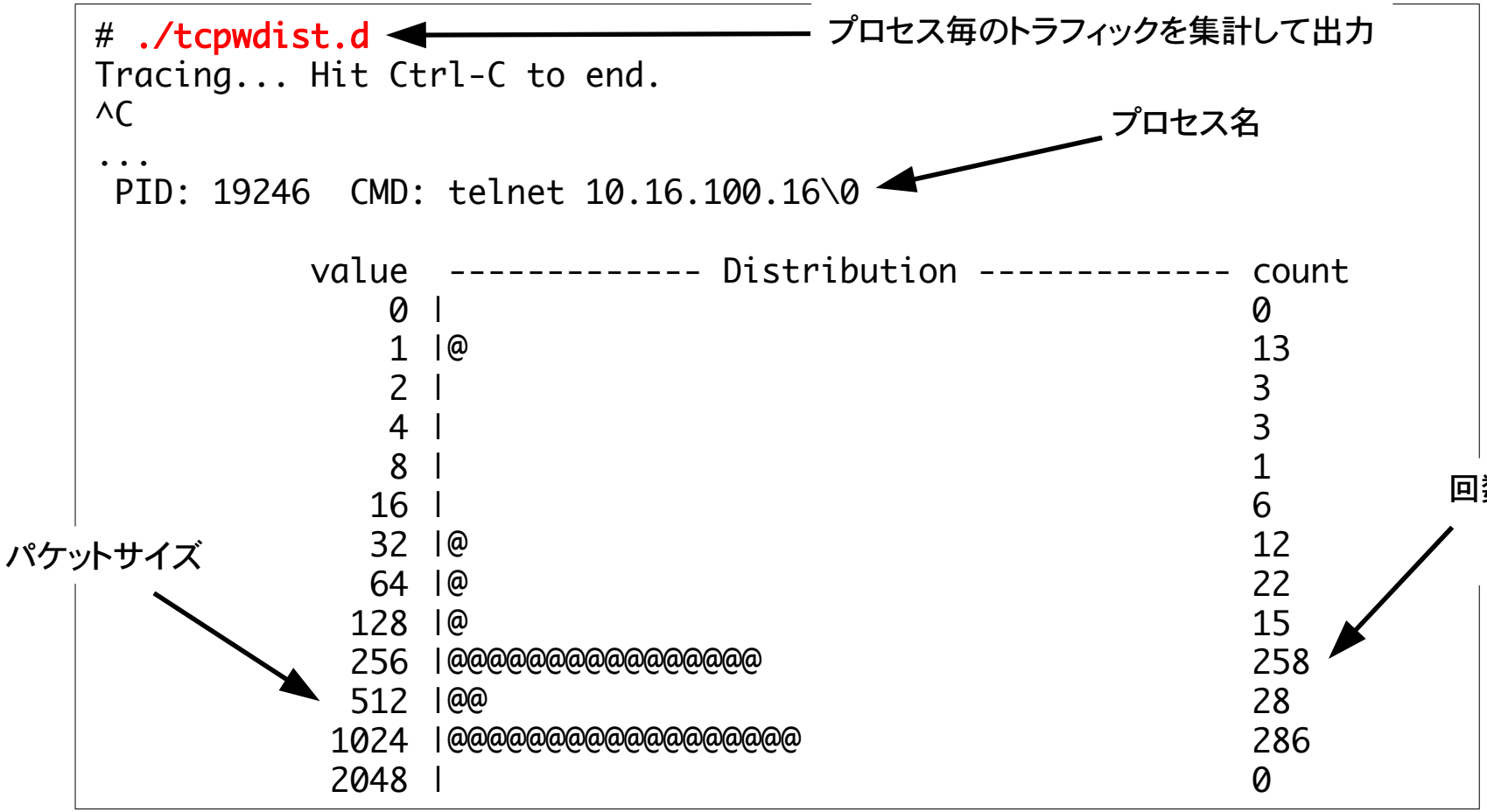
## プロセス毎のトラフィックを調べる

~~ それぞれのプロセスがどんなトラフィックを
発生させているかを調べます ~~

ORACLE

# Tips #16 プロセス毎のトラフィックを調べる

- DTrace Toolkit には tcpwdist.d という、プロセス毎に TCP の送受信サイズの分布を調査するスクリプトが入っています。




DTrace Toolkit : http://opensolaris.org/os/community/dtrace/dtracetoolkit/

# おわりに

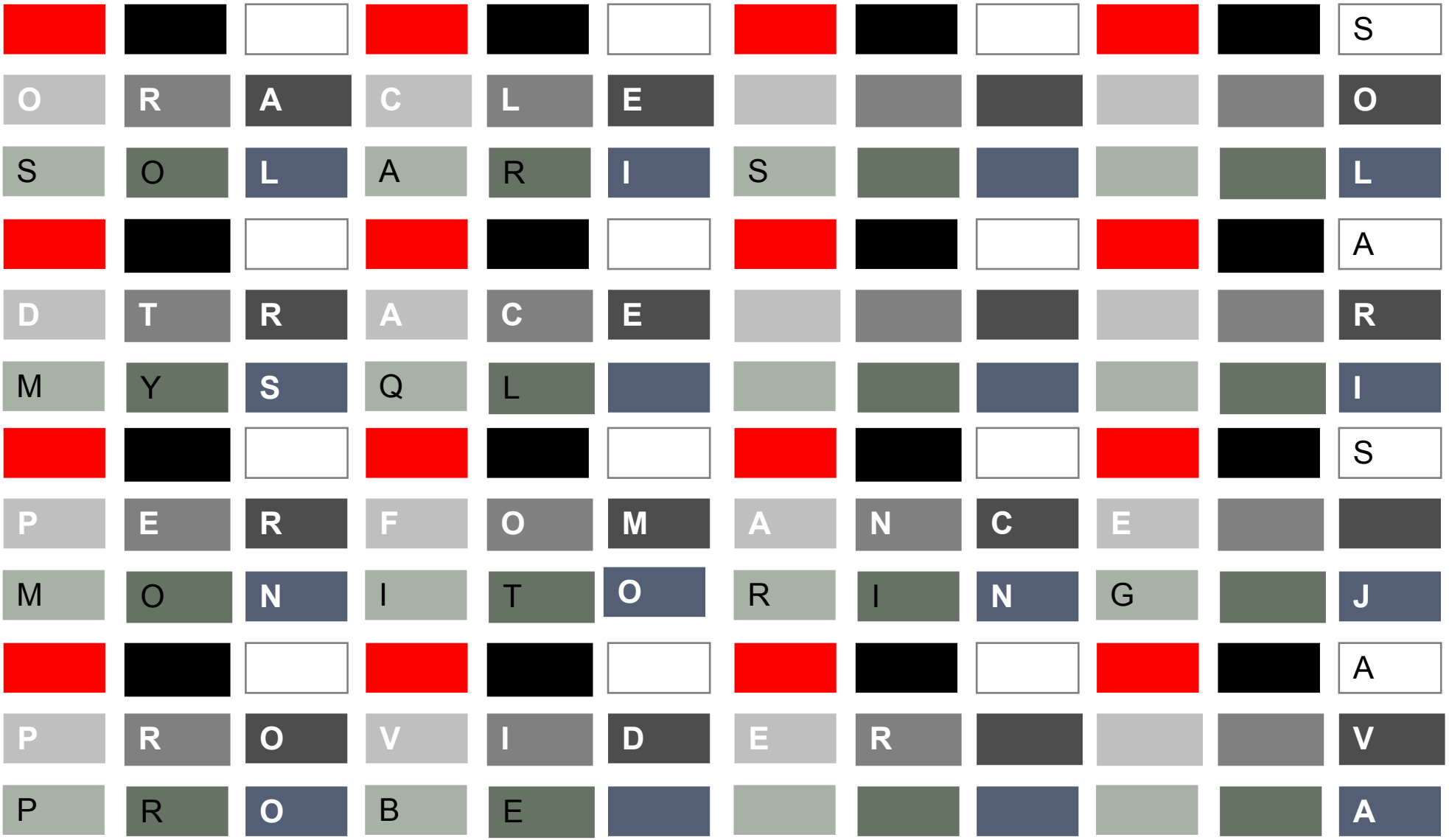
- 以上 Solaris のネットワーク解析ツールの内、基本的な物をご紹介しました。状況に応じて各コマンドを使い分け、問題解決にお役立て下さい。DTrace や kstat を使用すると、更に詳細な分析を行う事も可能です。是非ご活用下さい。

# ネットワークチューニングのヒント

- 秘訣は、テスト環境を作ってどんどん試す事

- Solaris Internals Siwiki
  - http://www.solarisinternals.com/wiki/index.php/Networks
- spec.org のレポートにあるチューニング例
  - http://www.spec.org/web2005/results/res2009q4/web2005-20091013-00143.html
- カーネルのチューンアップ・リファレンスマニュアル
  - http://docs.sun.com/app/docs/doc/819-0376
- パラメータの意味は実装を読むのが一番です
  - http://src.opensolaris.org/

# ご清聴ありがとうございました



あなたにいちばん近いオラクル

# Oracle Direct

## まずはお問合せください

Oracle Direct　検索

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。

システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

### Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。

http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力には、Oracle Direct Seminar申込時と同じ
　ログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので、ご登録さ　れ
ている連絡先が最新のものになっているか、ご確認下さい。

### フリーダイヤル

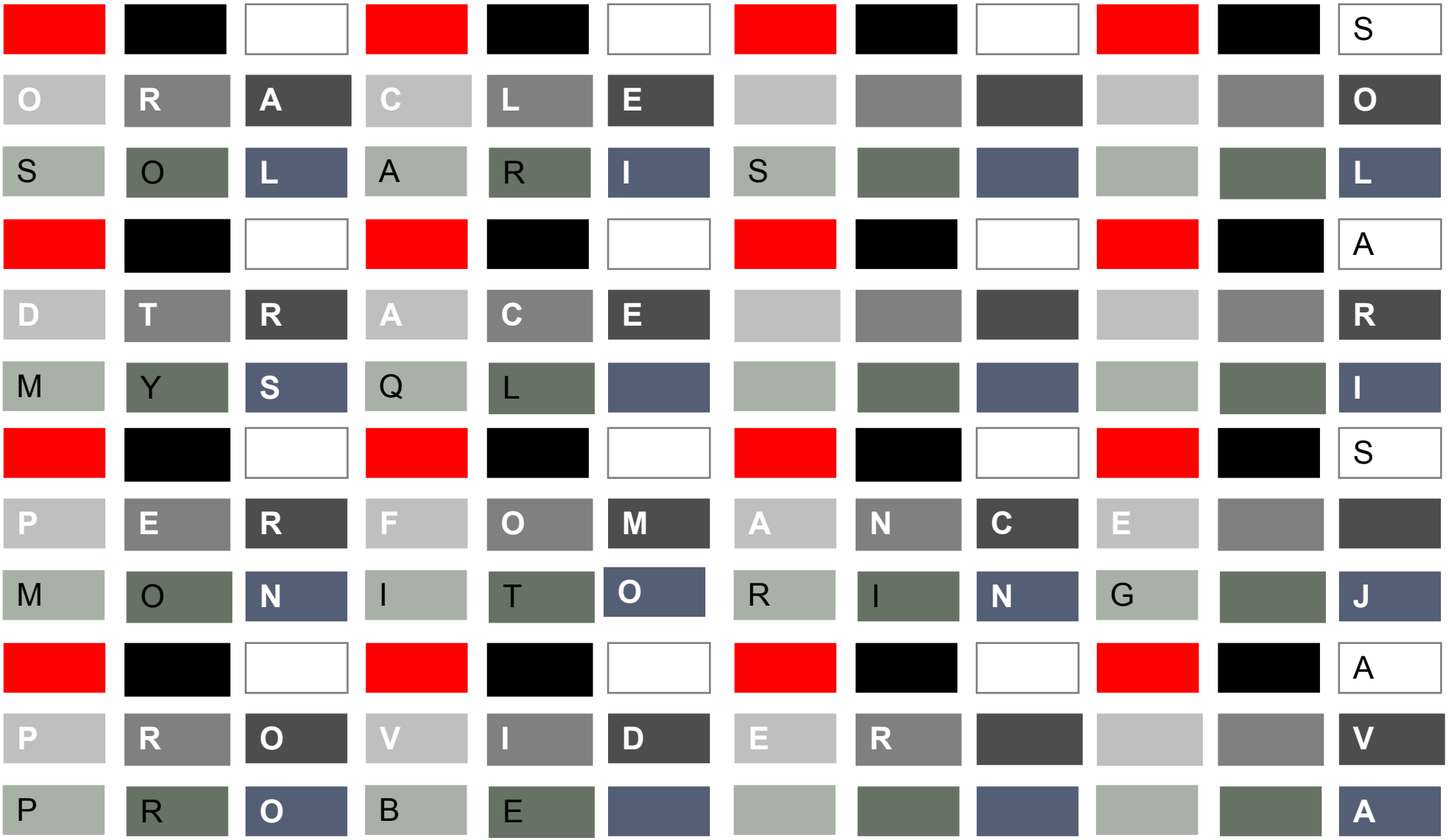
**0120―155―096**

※月曜~金曜 9:00~12:00、13:00~18:00
（祝日および年末年始除く）

**SOFTWARE. HARDWARE. COMPLETE.**


ORACLE®

# Backup Slides



# スライドのタイトルを記入

- First-level bullet
  - Second-level bullet
    - Third-level bullet
      - Fourth-level bullet
        - Fifth-level bullet
        -
        -