

SpringOS 概要

StarBED Project

情報通信研究機構 北陸リサーチセンタ

Hokuriku Research Center,

National Institute of Information and Communications Technology

分散処理環境 SpringOS

- 多数ホスト上のプロセス起動、その順序制御
- ネットワーク実験向け
- 各種自動設定
 - ◇ 電源投入・切断、起動OS切り替え
 - ◇ ネットワーク (VLAN) 構築
- ディスクイメージ収集・配布
- スクリプト言語でプログラミング可能
 - ◇ 役割毎に処理を記述
 - ◇ メッセージ交換、バリア同期

使い方

- スクリプトを実行
 - ◇ これが主流
 - ◇ バッチ的使い方
- シェルで逐次実行
 - ◇ スクリプトができるまでの準備
 - ★ ディスクイメージを作る
 - ◇ スクリプトを使わない
 - ★ 電源投入だけ
 - ★ ネットワーク構築だけ

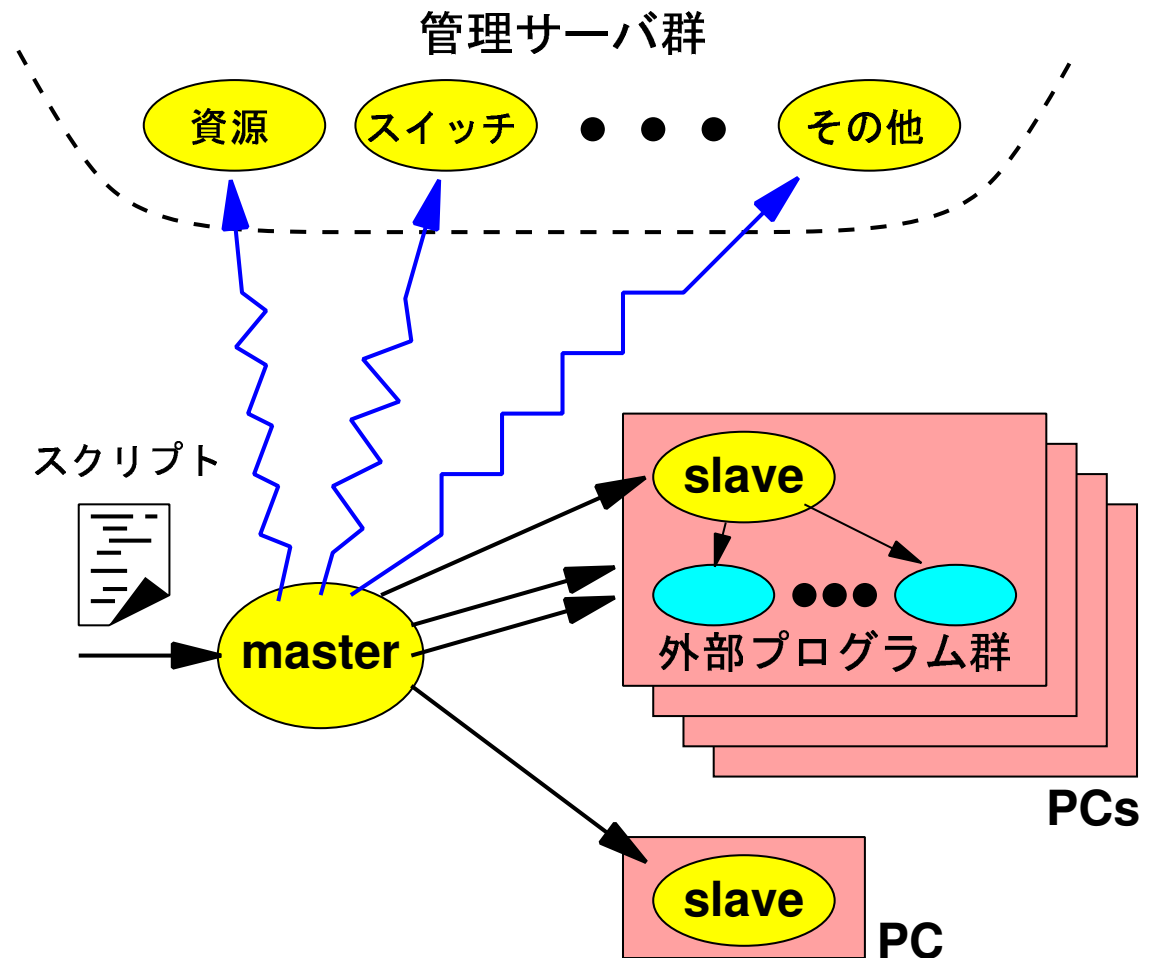
構成 (マクロ)

スクリプト評価

- master
 - ◇ スクリプト分配
 - ◇ ローカル評価
- slave
 - ◇ リモート評価

管理サーバ群

- 資源、スイッチ、起動...



★システム構成 注目点

- いくつかの小さなモジュールに分割
- モジュール間は TCP で情報交換
- 一般的な機構を多数利用
 - ◇ DHCP, TFTP, PXE, FTP
- 存在しない、あるいは効率が悪い箇所は独自開発
 - ◇ 評価器、スイッチ制御、資源管理

特徴: ネットワーク実験向け

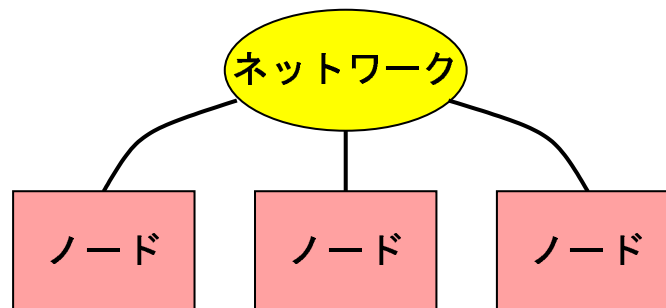
- 多数のノードを同時制御
 - ◇ 最大実績 300台程度、限界未到達
- MACアドレスやスイッチポートの情報を保持
- IPアドレス自動決定
- 記述に合わせて各種スイッチを制御
 - ◇ IOS, Ironware, OSL2, D-Link 等に対応

記述言語: K言語 (コードネームそのまま)

- スクリプト言語
 - ◇ 原則、1行1構文 // シェルに近い
- ノードとネットワークを主体に記述
- 宣言文と実行文が混在
- 宣言が一段落したら一括設定
 - ◇ ノード獲得
 - ◇ ディスクイメージ配布
 - ◇ ネットワーク設定

K: ネットワーク宣言

- ネットワークはノード群で構成される



```
net corenet {  
    ipaddrange "192.168.12.0/24"  
}
```

- IPアドレス範囲を指定（自動割当のヒントに）
- ノードの参加は別構文（後述）

K: ノード宣言

```
node client {
  netif media gigabitethernet
  netif media gigabitethernet
  ...
  scenario {
    ...
  }
}
```

- 必要に応じてノード仕様の条件を指定
 - ◇ ネットワークや起動方法等
- ノードの行う処理をシナリオ（後述）として記述

K: ネットワーク参加

attach 構文

- N/W I/F 毎に参加ネットワークを登録

```
attach client.netif[0] corenet  
attach client.netif[1] leafnet
```

具体的な設定は各ノードで ifconfig 等呼び出す

- I/F 識別は MAC アドレスで
- MAC アドレスとデバイス名を結びつけるツールも用意

K: シナリオ

実行文列を「シナリオ」と呼ぶ

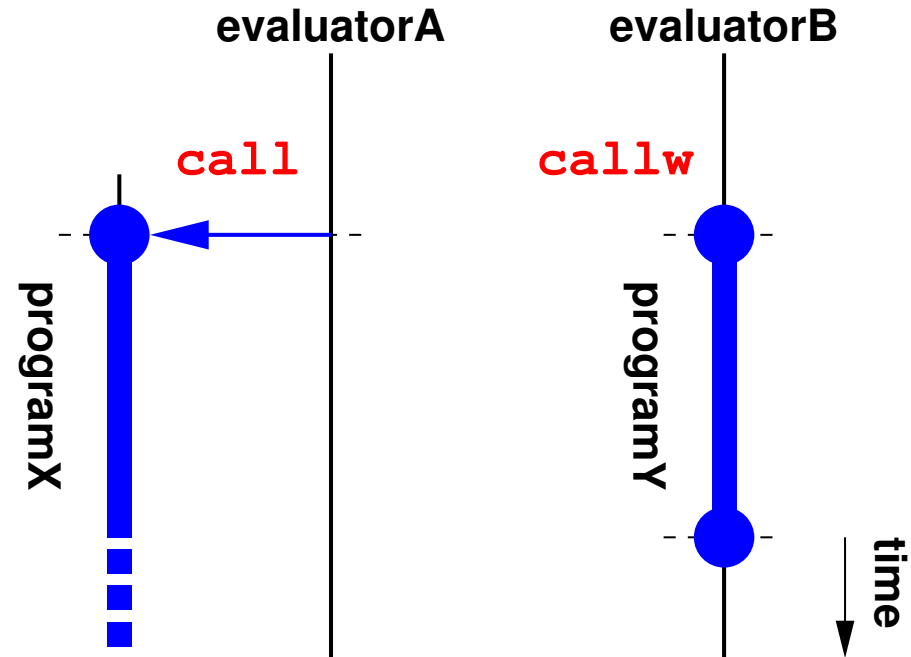
- master 側 (ローカル) = グローバルシナリオ
- slave 側 (リモート) = ノードシナリオ

主な機能

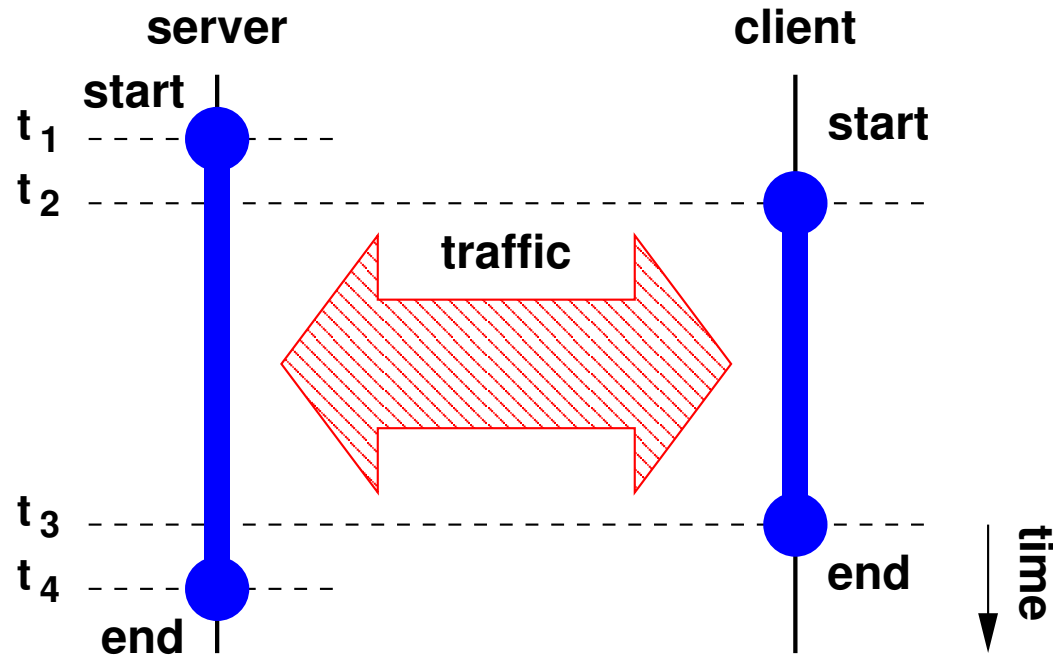
- 外部プログラム起動
- メッセージ交換、同期
- ループ
- ファイル転送

外部プログラム起動

- call 構文
 - ◇ 起動後そのまま
 - ◇ 並列
 - ◇ 分岐
- callw 構文
 - ◇ 起動後終了待ち
 - ◇ 直列



K: 記述例 クライアント・サーバ



- クライアント起動はサーバ起動の後
- サーバ停止はクライアント終了の後

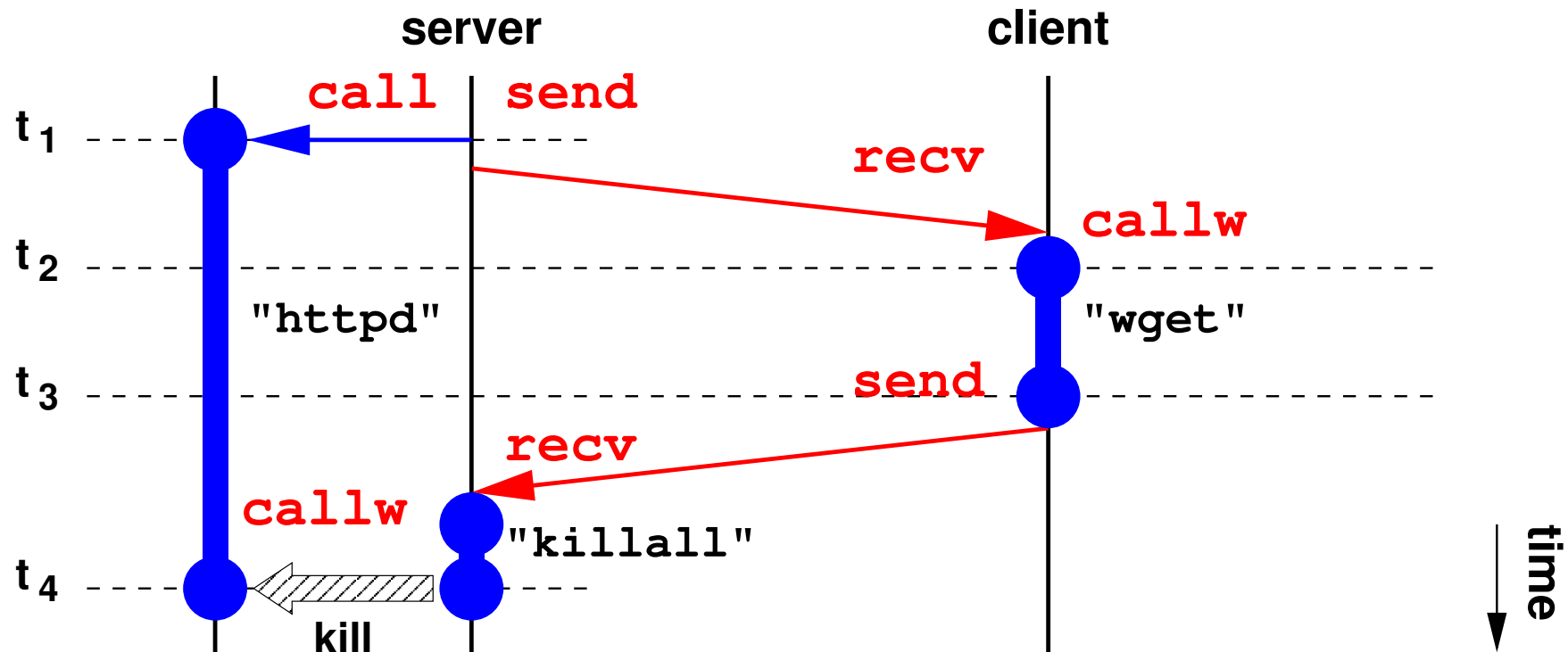
K: 記述例 クライアント・サーバ (*cont.*)

メッセージ交換で順序を記述

- 受信待ちで処理中断 (ブロック)
- 受信すると処理再開

server	client
<code>call "httpd"</code>	
<code>send "server-up"</code>	
	<code>recv msg</code>
	<code>callw "wget"</code>
<code>recv msg</code>	
<code>callw "killall" "httpd"</code>	<code>send "client-done"</code>

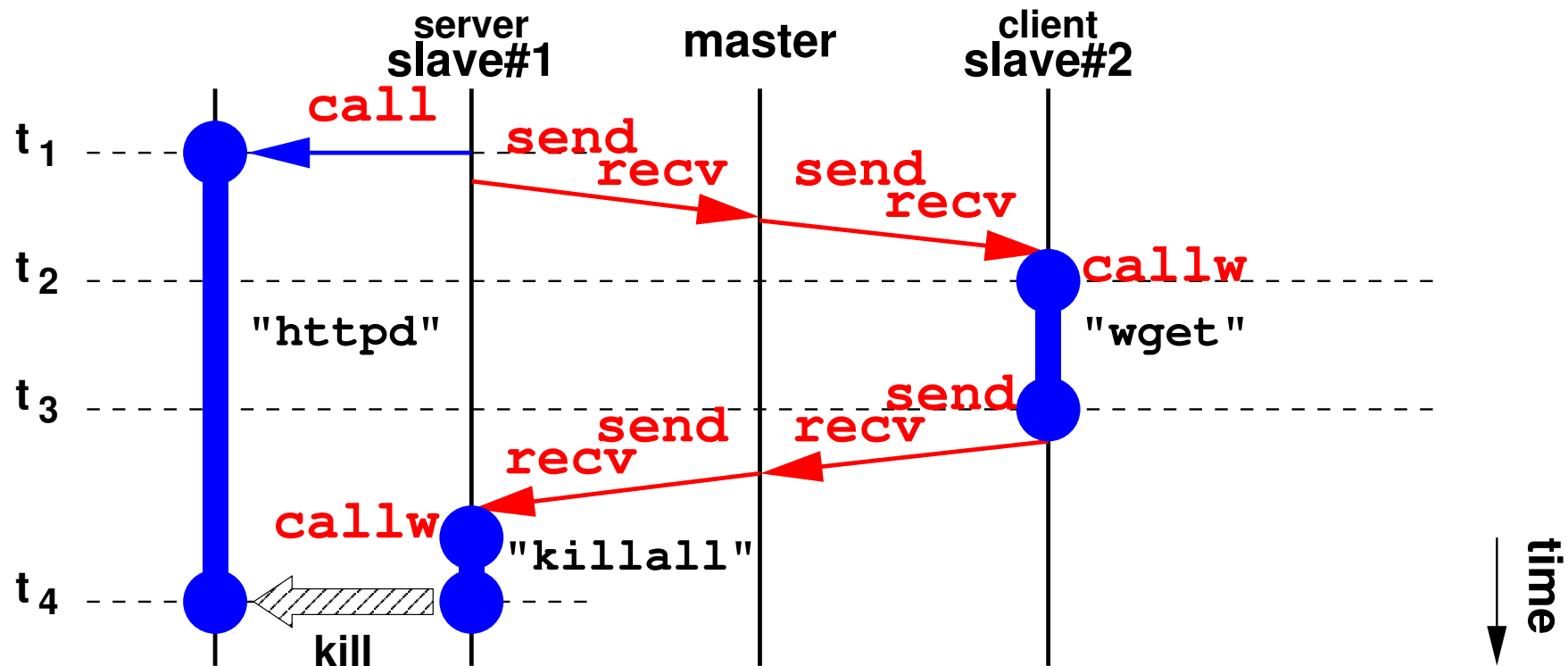
K: 記述例 クライアント・サーバ (*cont.*)



所定の順序を表現できた。

★master の必要性

slave は互いを知らないので master が中継



★master の必要性 (cont.)

master のシナリオ

```
recv server msg
send client msg
recv client msg
send server msg
```

ノードが増えてくると全体を見渡した処理が必要

- 全体通知、全体待機

全体処理は master の役割

K: ノードクラス

- 同じ構成のノードを多数利用する場面が多い
- 役割毎にノードを記述して手間と間違いを軽減

```
nodeclass wwwclient {  
    ...  
    scenario {  
        ...  
    }  
}  
N=30  
nodeset A class wwwclient num N
```

★K言語: 趣味の領域?

名前空間

- 原則、評価器毎に名前空間は別
 - ◇ 特定宣言で master から slave へ変数継承

実装

- 内部は LISP、データの内部表現は S 式
- master と slave の通信は一種の S 式ストリーム
- あまり凝った実装ではない
- lambda 関数程度は扱える

★工夫: 資源管理

- (予約を考慮する)
- 条件に合わせて資源を割り当てる
- 排他処理で衝突を回避
- MAC アドレスやスイッチポート番号等の情報を保持

できない事

- L2、L1 は扱えない // 対象外
- WINDOWS は扱えない // 対応未定
- タグつき VLAN は扱えない // 対応検討中

仮想化技術への対応

仮想機械を扱う

- Parallels 上で動作
 - ◇ (仕様により) 電源管理と PXE 起動ができない
- VMwere 上で動作
 - ◇ (仕様により) 電源管理ができない
- Xen は未確認

仮想環境全体を直接駆動

- 旧バージョンは VMware を駆動可能

動作実績

- StarBED を想定しているが他の施設でも動く可能性は高い
- StarBED 外で数箇所、動作実績あり
- WIDE nerdbox WG で普及活動開始

参考文献

[1] StarBED Home Page <http://www.starbed.org/>

[2] 「できる StarBED」

[3] 「K言語リファレンスマニュアル」

お問い合わせは info@starbed.org へ