



SpringOS/VM:

大規模ネットワークテストベッドにおける 仮想機械運用技術

宮地利幸, 知念賢一, 篠田陽一

toshi-m@jaist.ac.jp.

北陸先端科学技術大学院大学

StarBED Project(<http://www.starbed.org/>)

- **実環境との同一性をもつ実ノードによる実験環境が求められている**
 - **実環境へ導入する前の実装のバグも含めた最終チェック**
- **大規模な実ノードによる実験環境が提案されている**
 - Netbed (<http://www.emulab.net/>)
 - StarBED (<http://www.starbed.org/>)
- **既存の環境の拡張は困難**
 - **拡張にはさまざまなコストが必要**
 - **実験トポロジに必要なノード数は実験に依存**

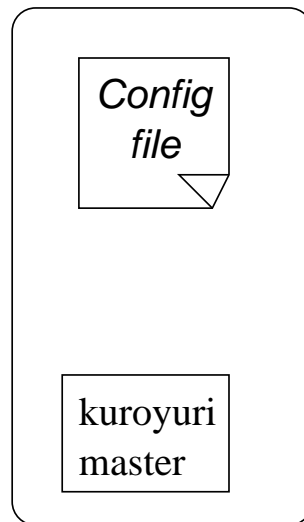
- 仮想ノードを用い実ノードを多重化
 - 様々なレベルで仮想ノードを実現する技術が発表されている
 - 汎用OSがそのままでは利用できない技術が多い
- 実ノードを用いた実験環境では実環境との同一性が重要視
 - 汎用OSを変更なしに利用できる仮想機械 (今回はVMware) を採用
 - 実験トポロジ中どの部分に仮想機械を利用できるかの検討は絶対必要

- 大規模な実験環境の各ノードの設定およびトポロジ設定、実験の遂行のコスト大
 - 実験支援システムによりサポート
 1. 実験遂行者の設定ファイルの読み込み
 2. リソースの割り当て
 3. ノードへのOSおよびアプリケーションの導入
 4. スイッチ設定による実験トポロジ構築
 5. シナリオの自動実行
 - 我々は実験支援システム SpringOS を開発中
 - SpringOS を拡張し仮想機械を扱う:SpringOS/VM

- 管理用ネットワークと実験用ネットワークの分離
- 管理用ネットワークのアドレスはDHCPによって自動設定
- 実験用ノードの情報は事前に登録
 - 各 I/F の MAC アドレス
 - 管理用ネットワークへ接続された I/F の IP アドレス
 - 各 I/F が接続されたスイッチおよびポート

SpringOS の処理手順

File
server



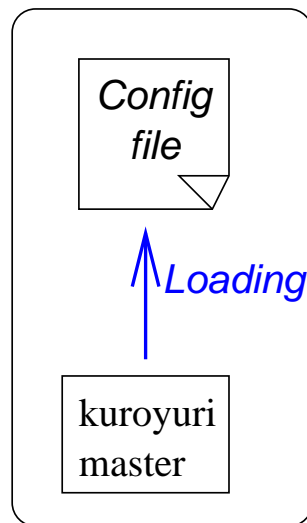
Management
node

erm

swconf

SpringOS の処理手順

File
server

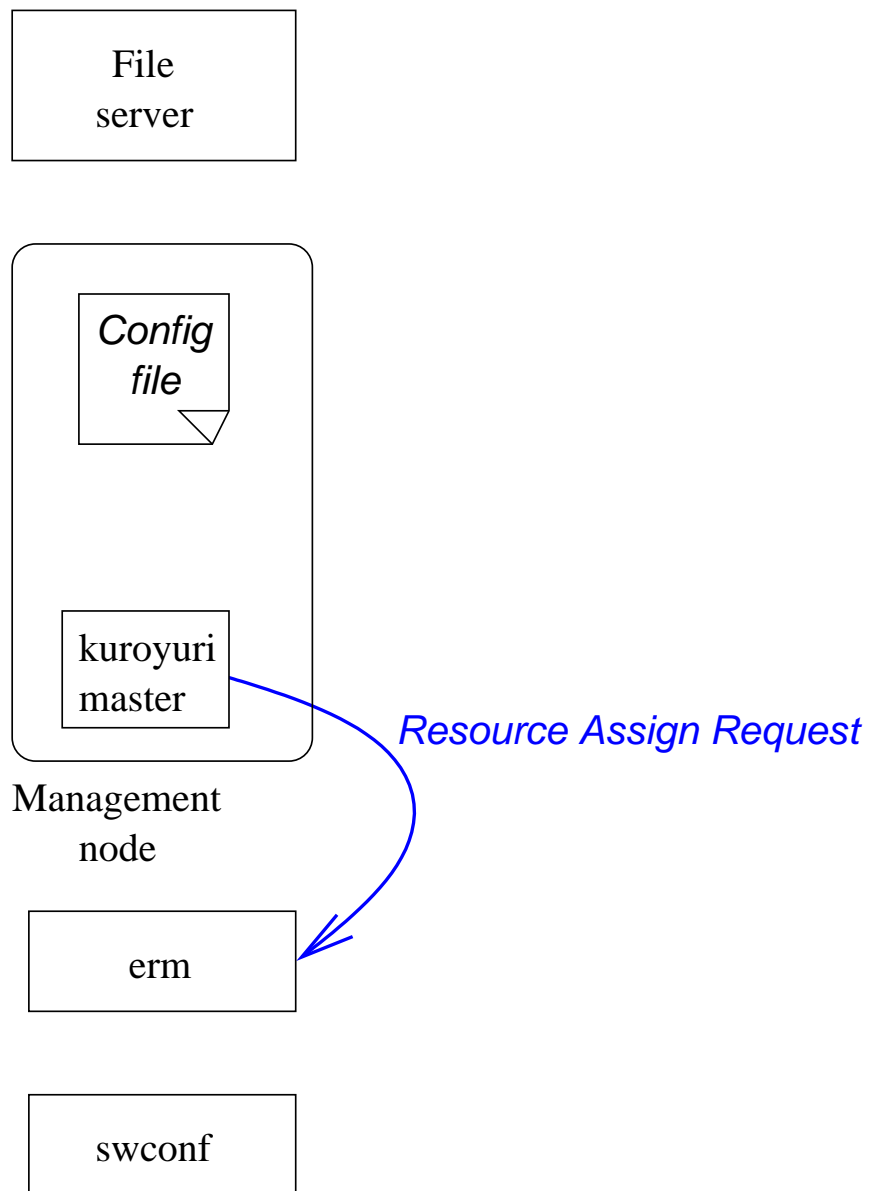


Management
node

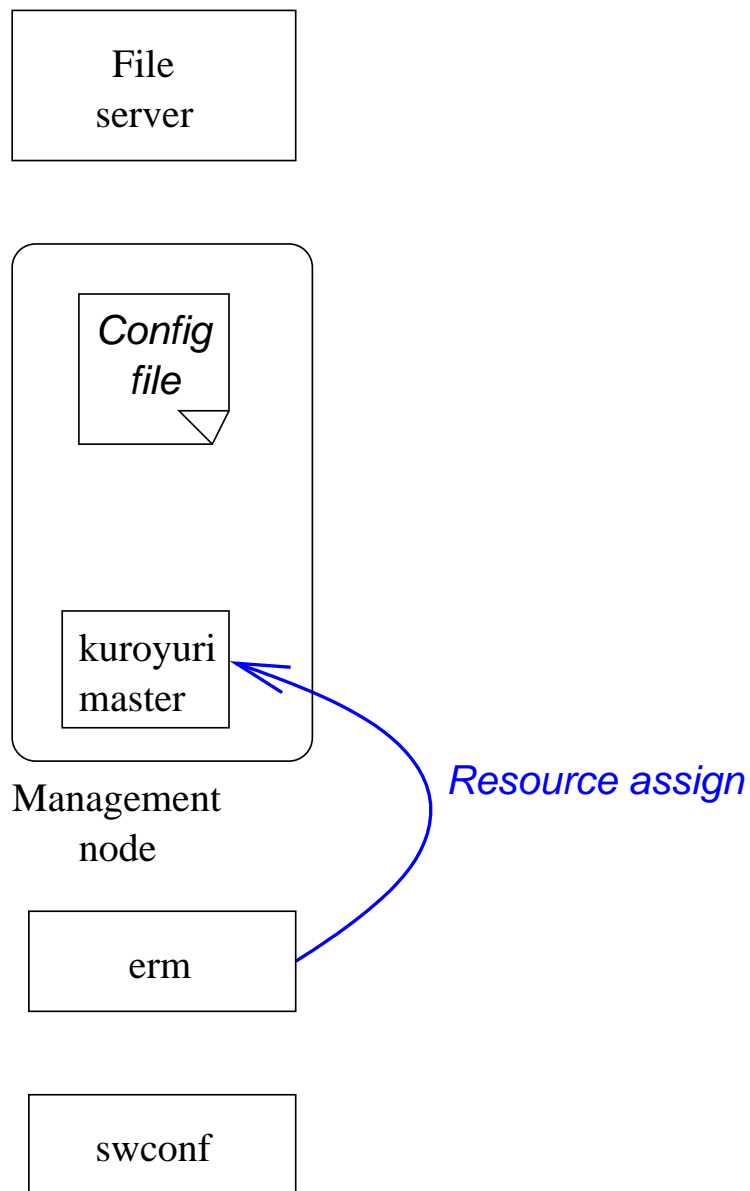
erm

swconf

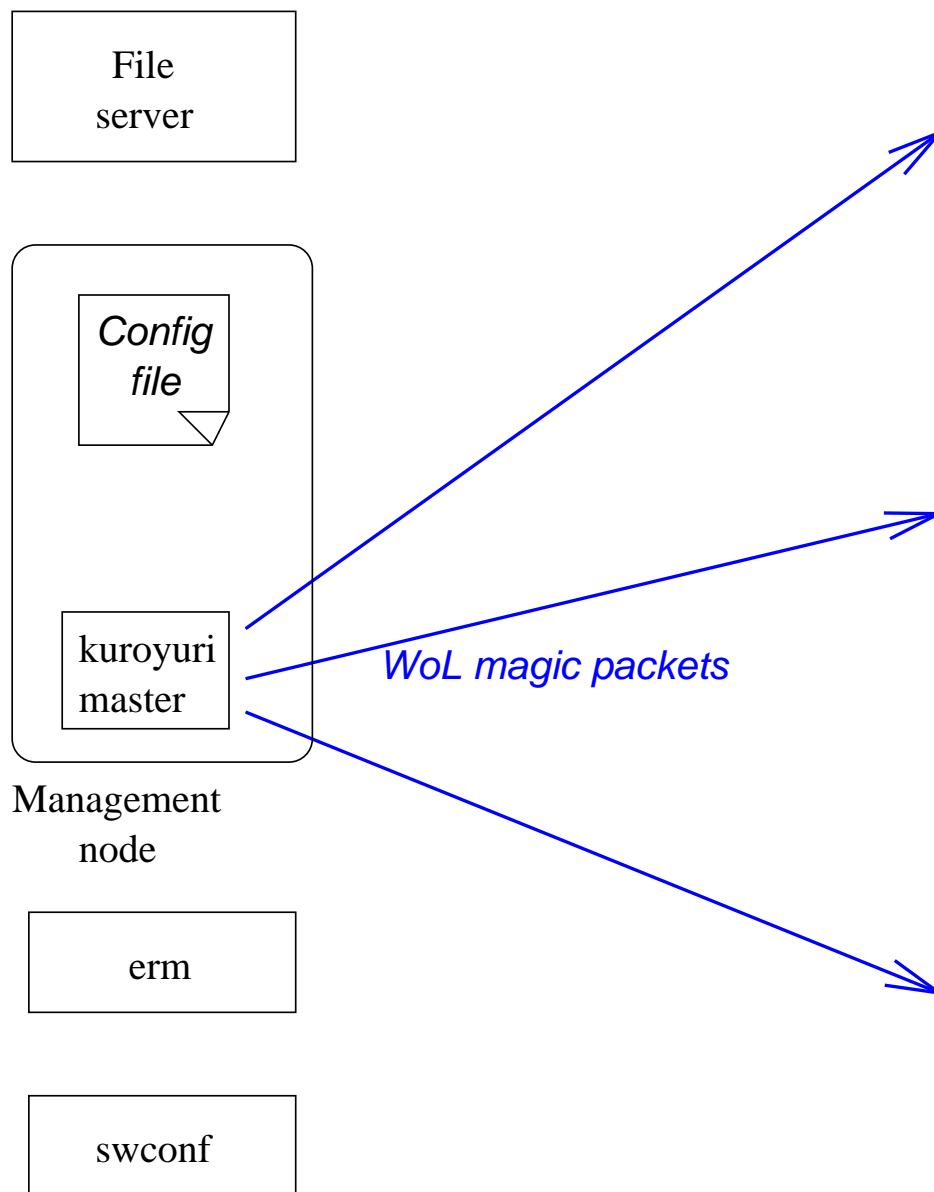
SpringOS の処理手順



SpringOS の処理手順

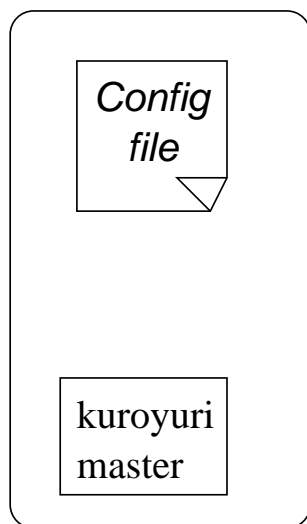


SpringOS の処理手順



SpringOS の処理手順

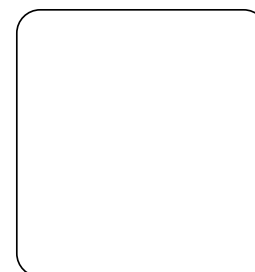
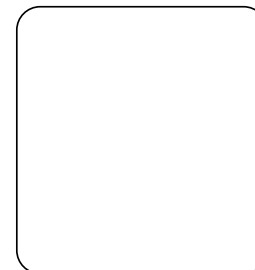
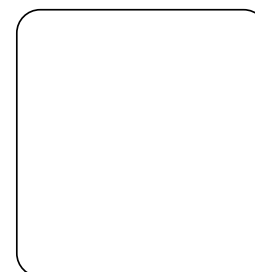
File
server



Management
node

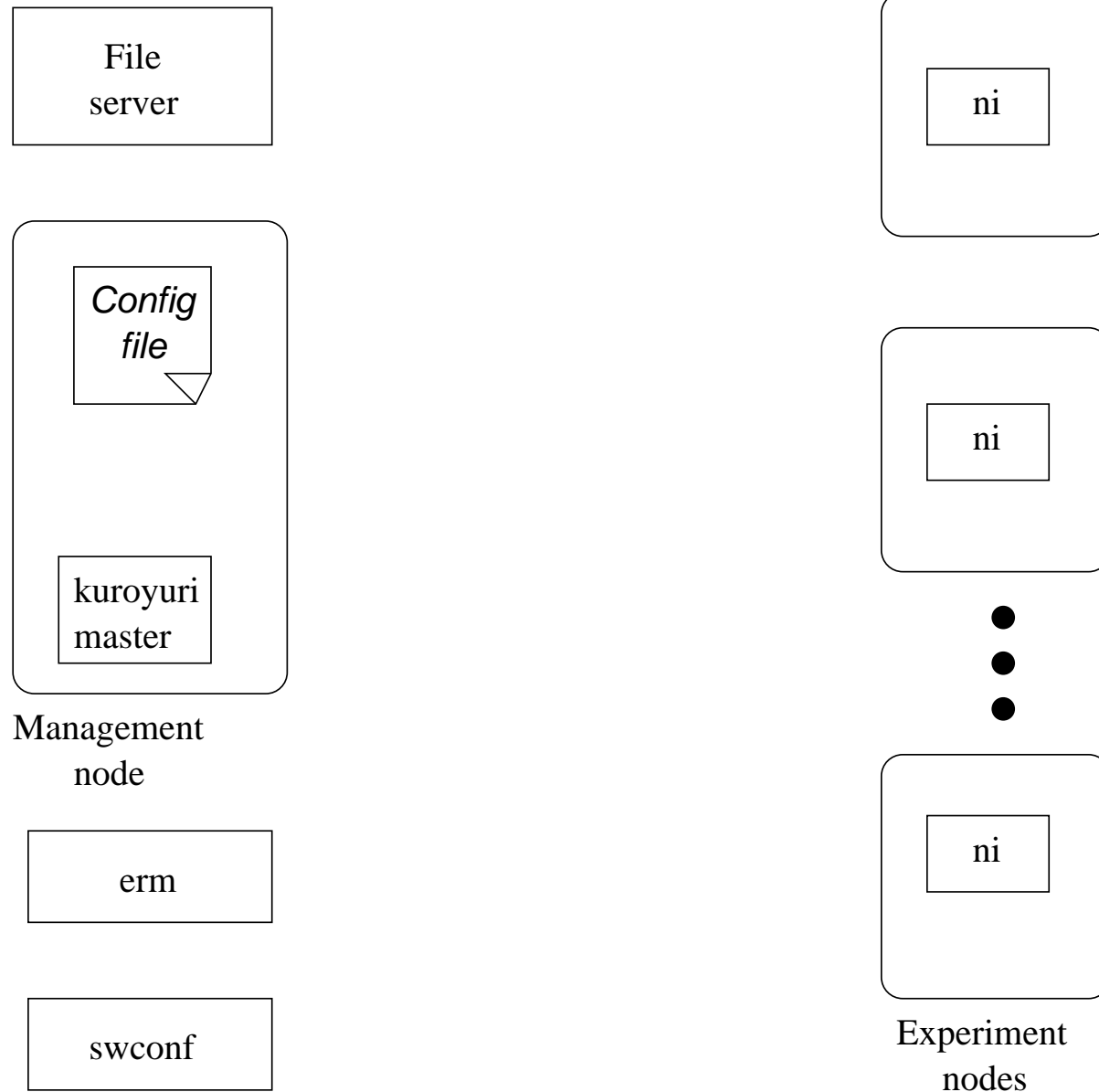
erm

swconf

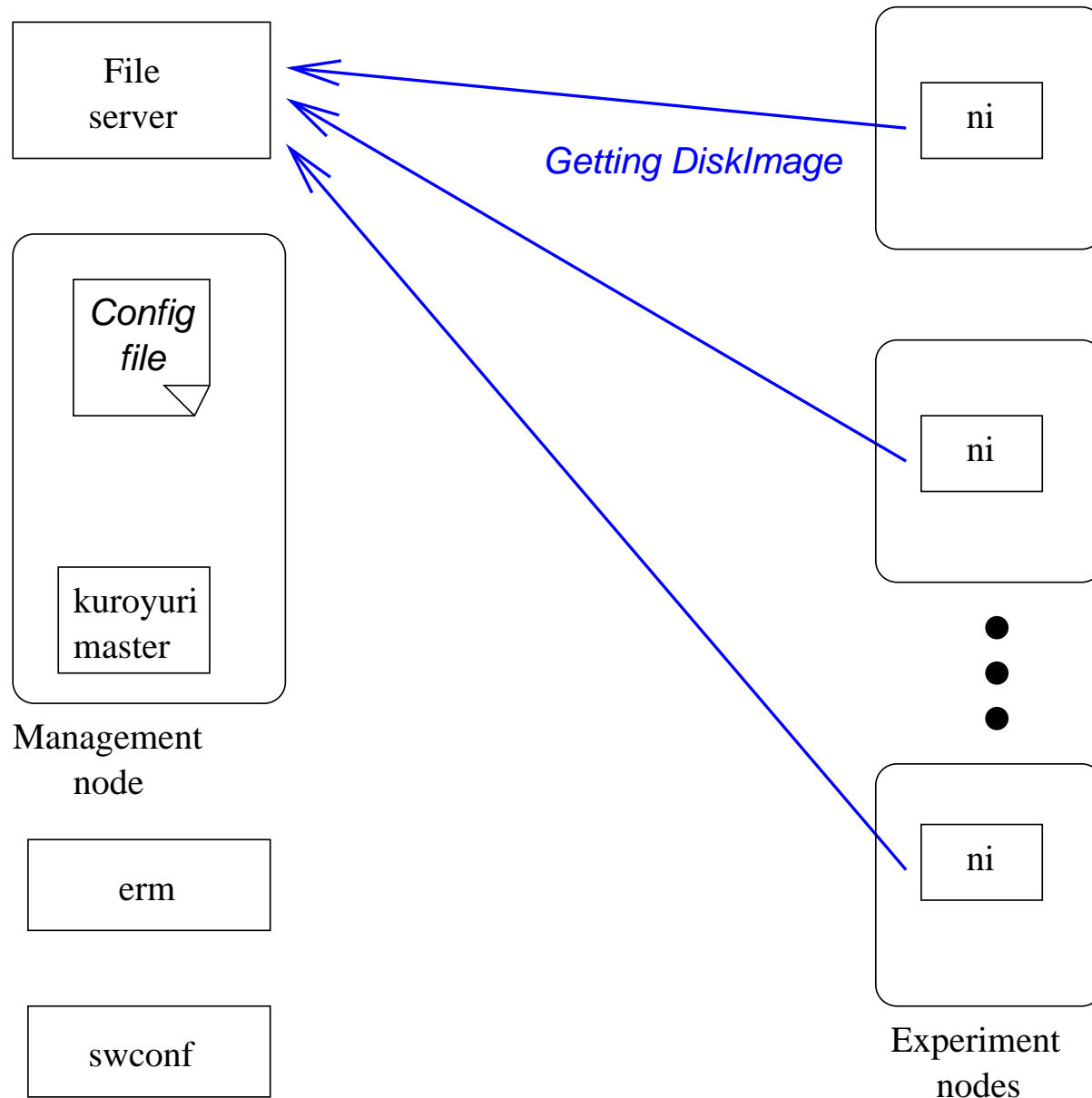


Experiment
nodes

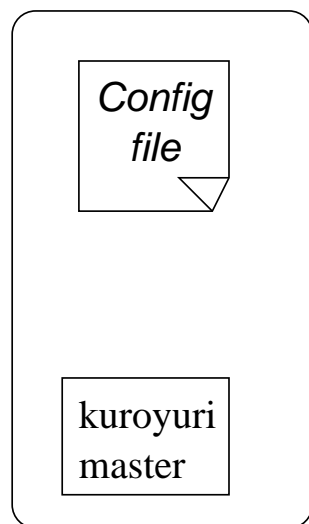
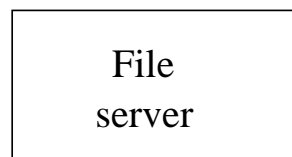
SpringOS の処理手順



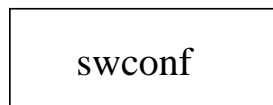
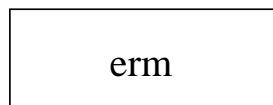
SpringOS の処理手順



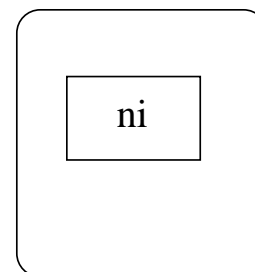
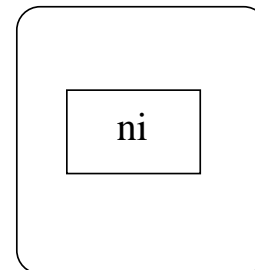
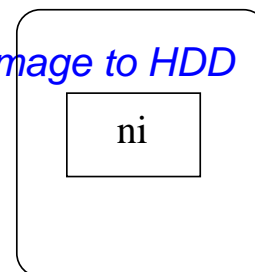
SpringOS の処理手順



Management
node

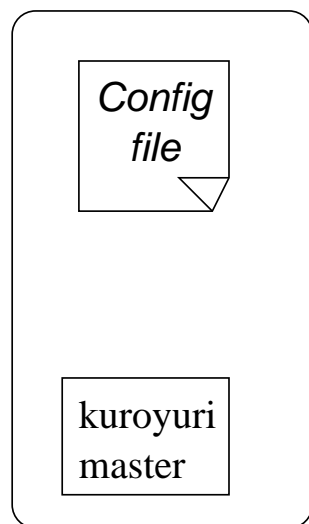
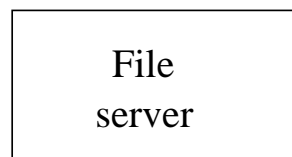


Writing DiskImage to HDD

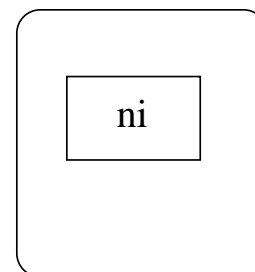
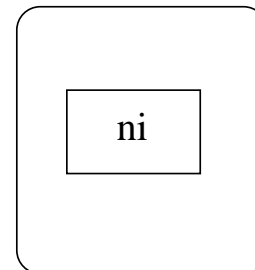
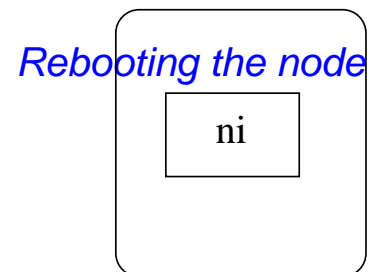
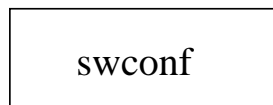
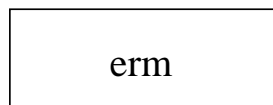


Experiment
nodes

SpringOS の処理手順

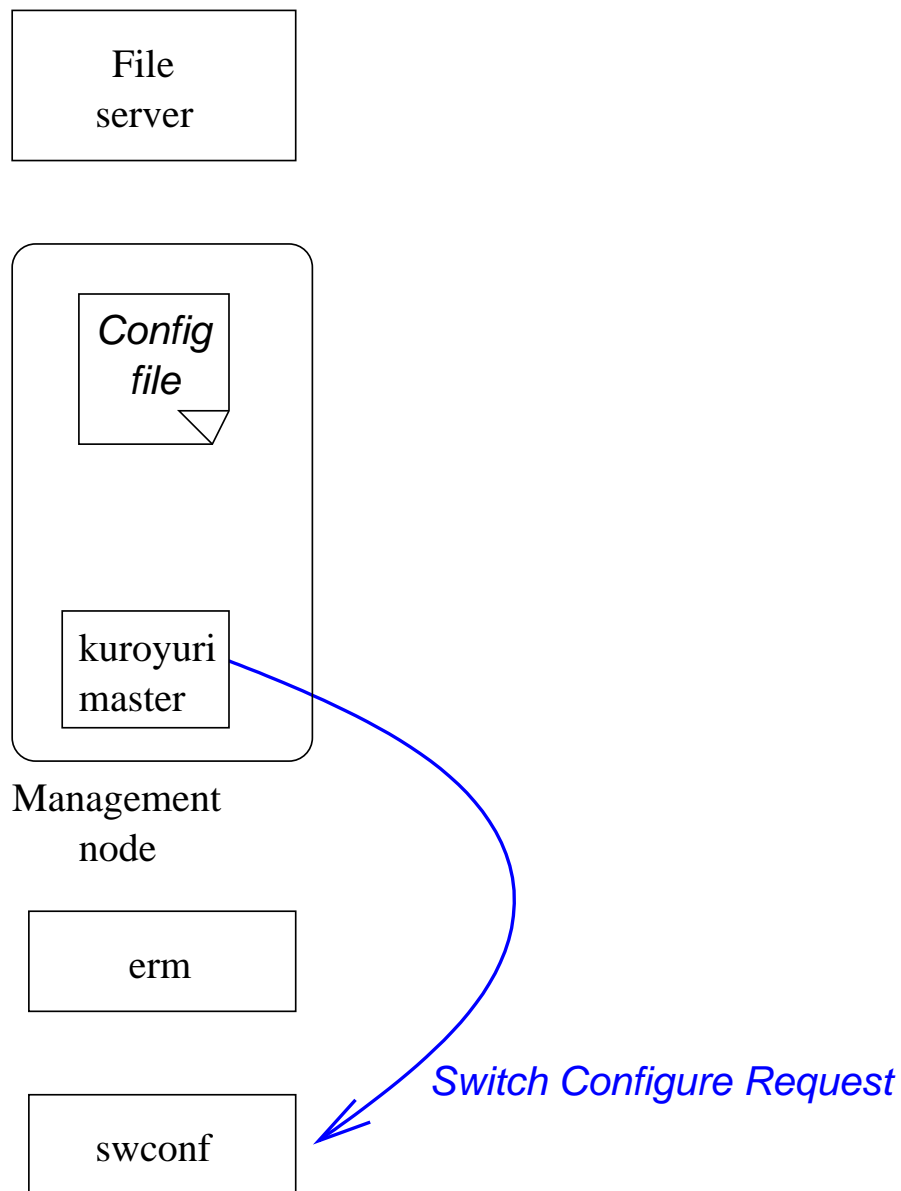


Management
node



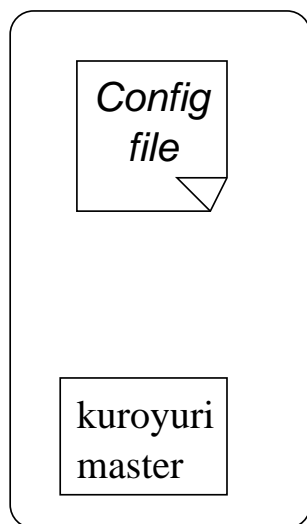
Experiment
nodes

SpringOS の処理手順



SpringOS の処理手順

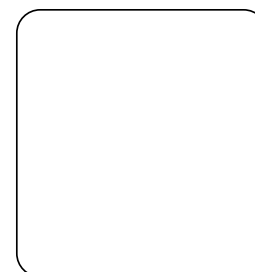
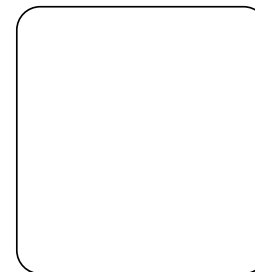
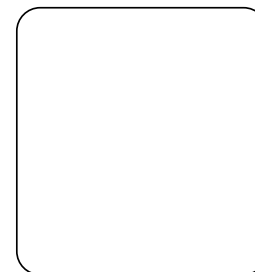
File
server



Management
node

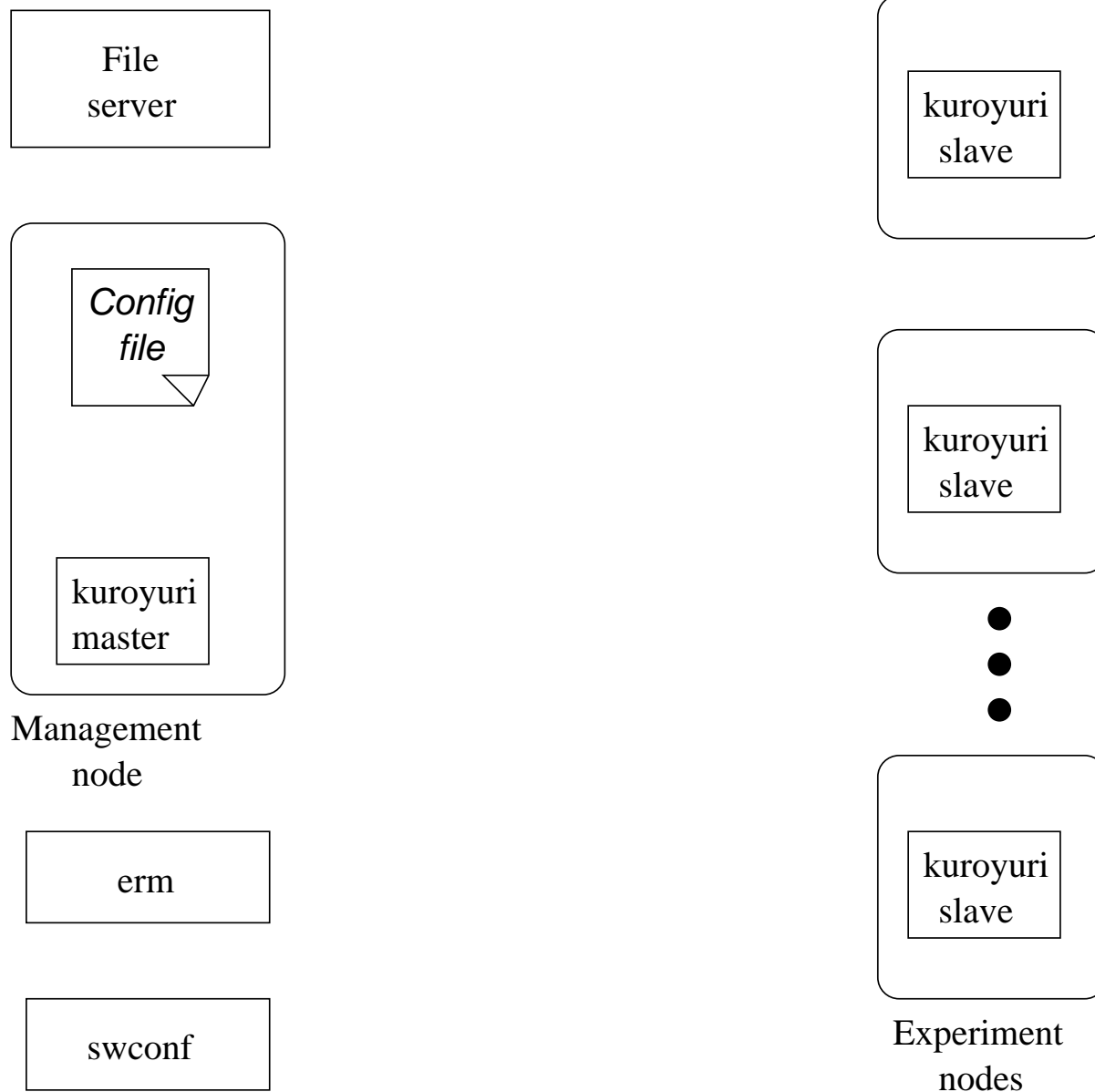
erm

swconf

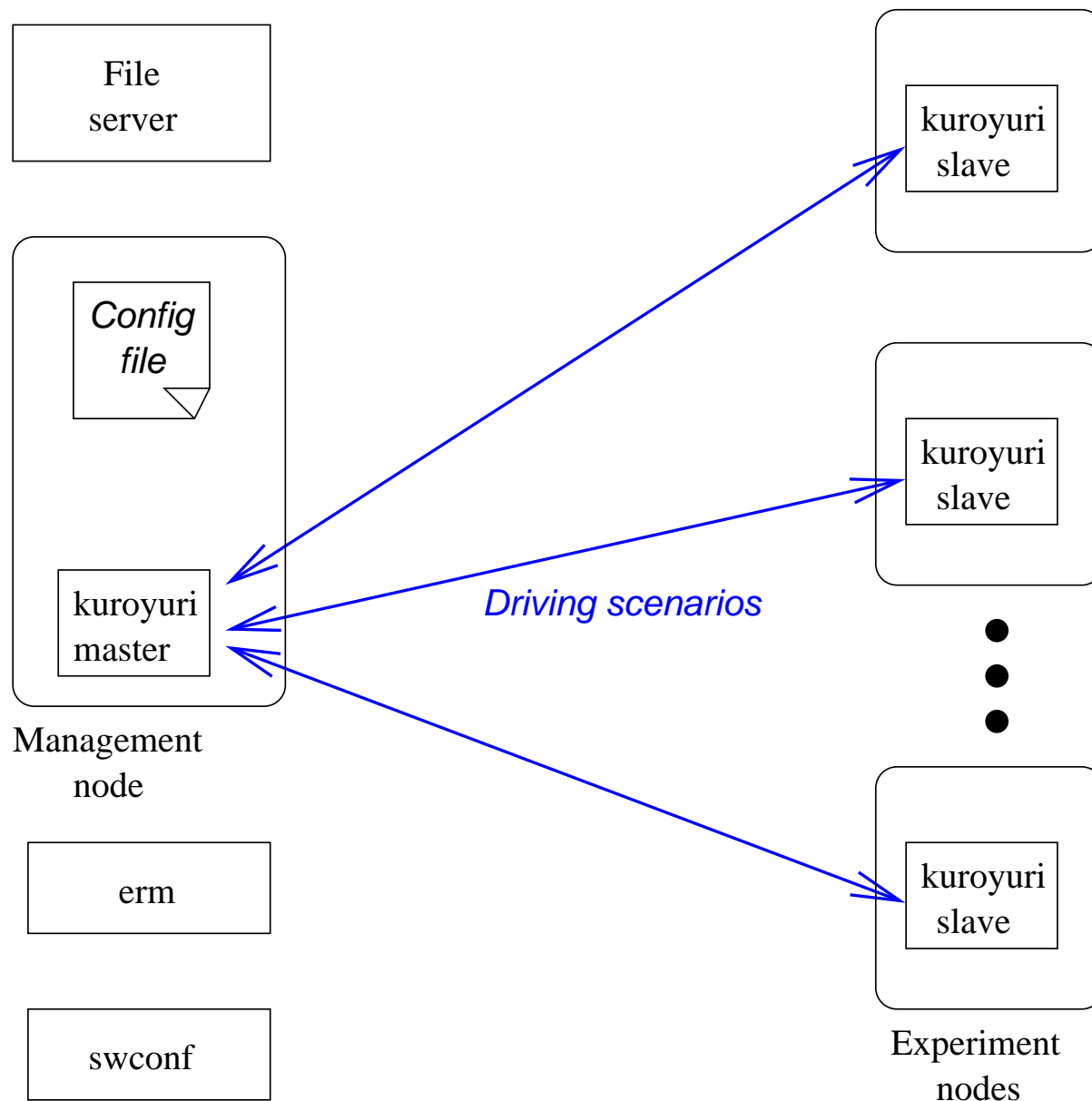


Experiment
nodes

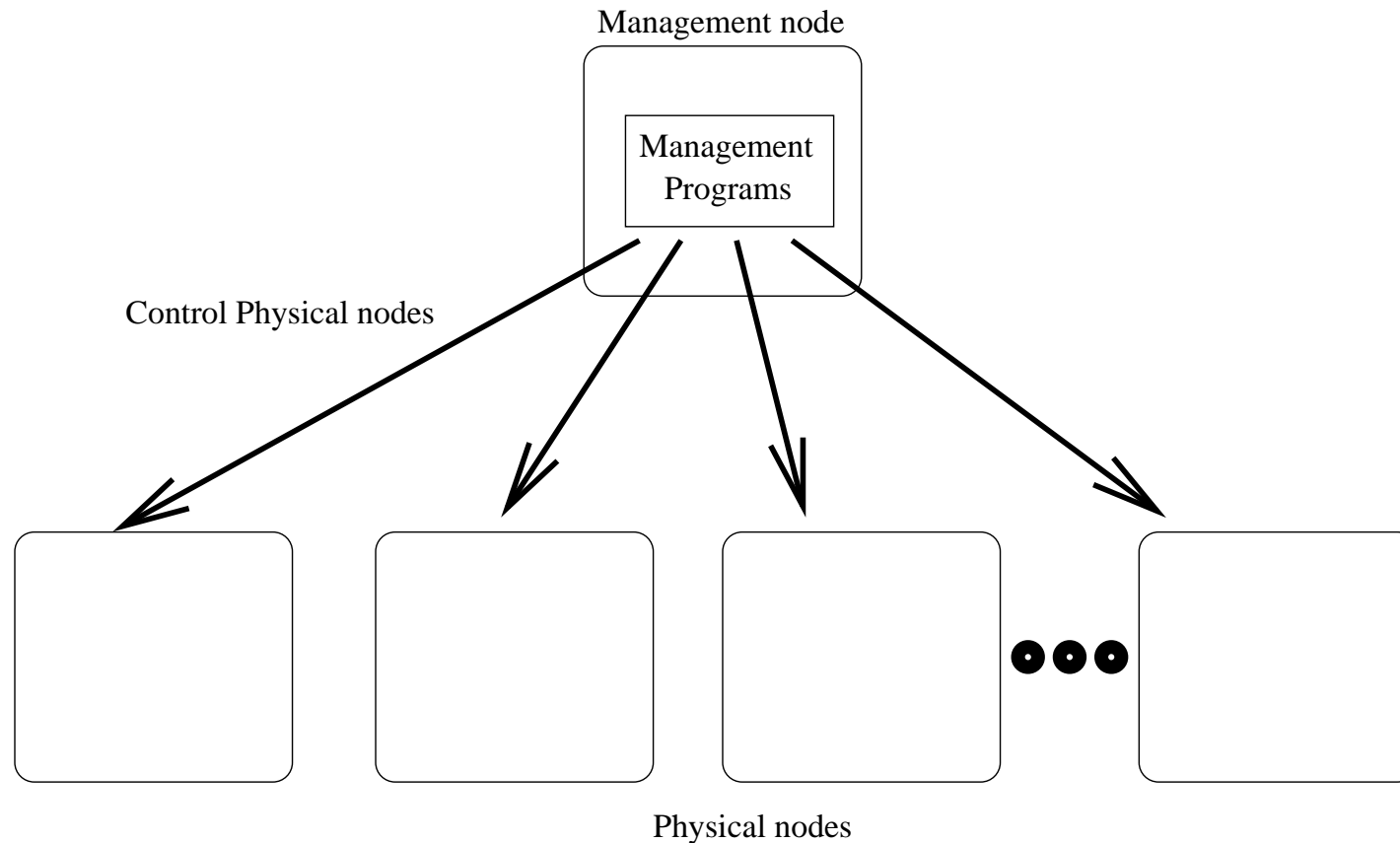
SpringOS の処理手順



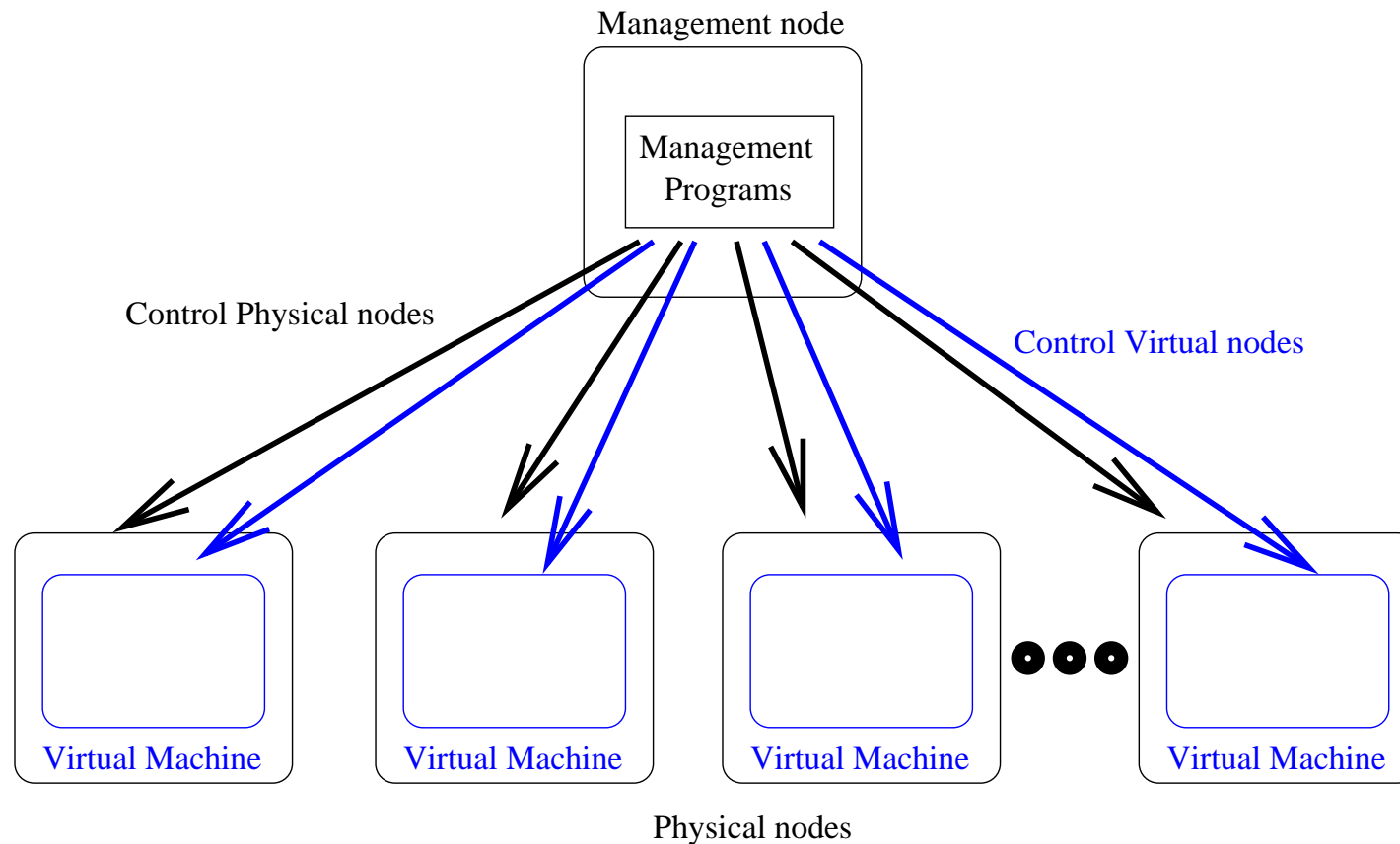
SpringOS の処理手順



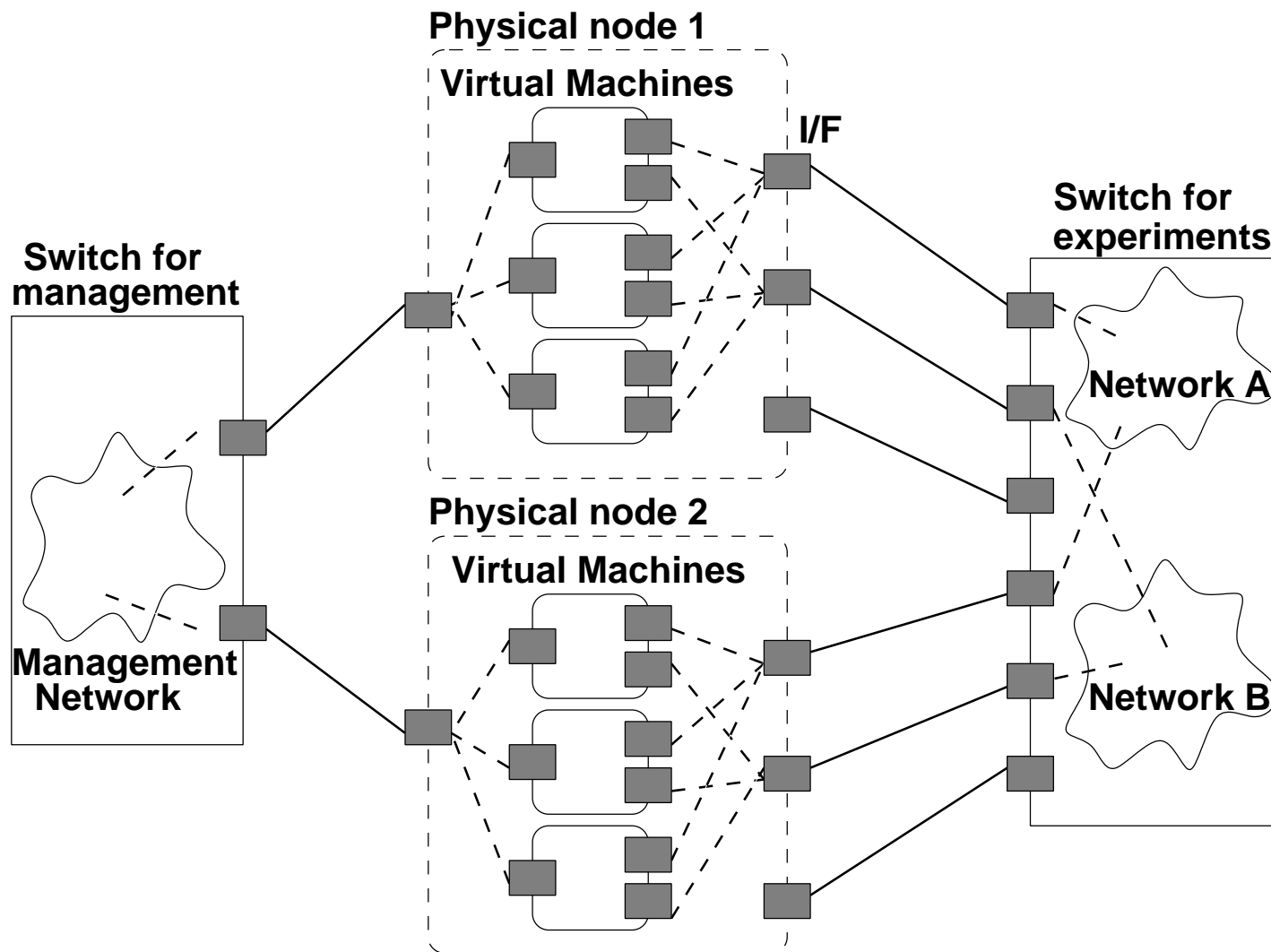
- 仮想機械を制御できるように SpringOS を拡張
 - 仮想機械のリソース制御
 - 仮想機械を動作させるノードと仮想機械の階層的制御



- 仮想機械を制御できるように SpringOS を拡張
 - 仮想機械のリソース制御
 - 仮想機械を動作させるノードと仮想機械の階層的制御

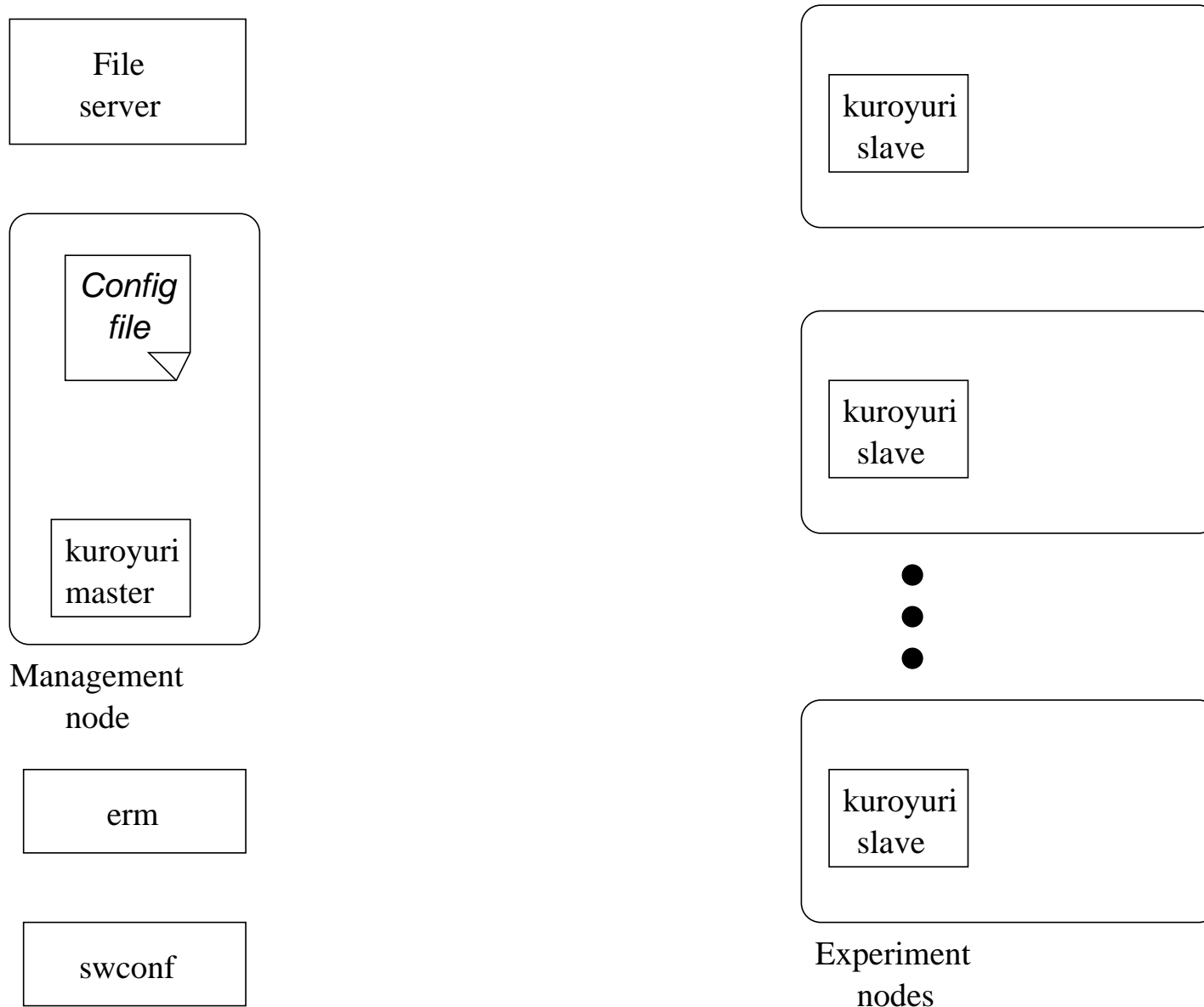


仮想機械の接続形態

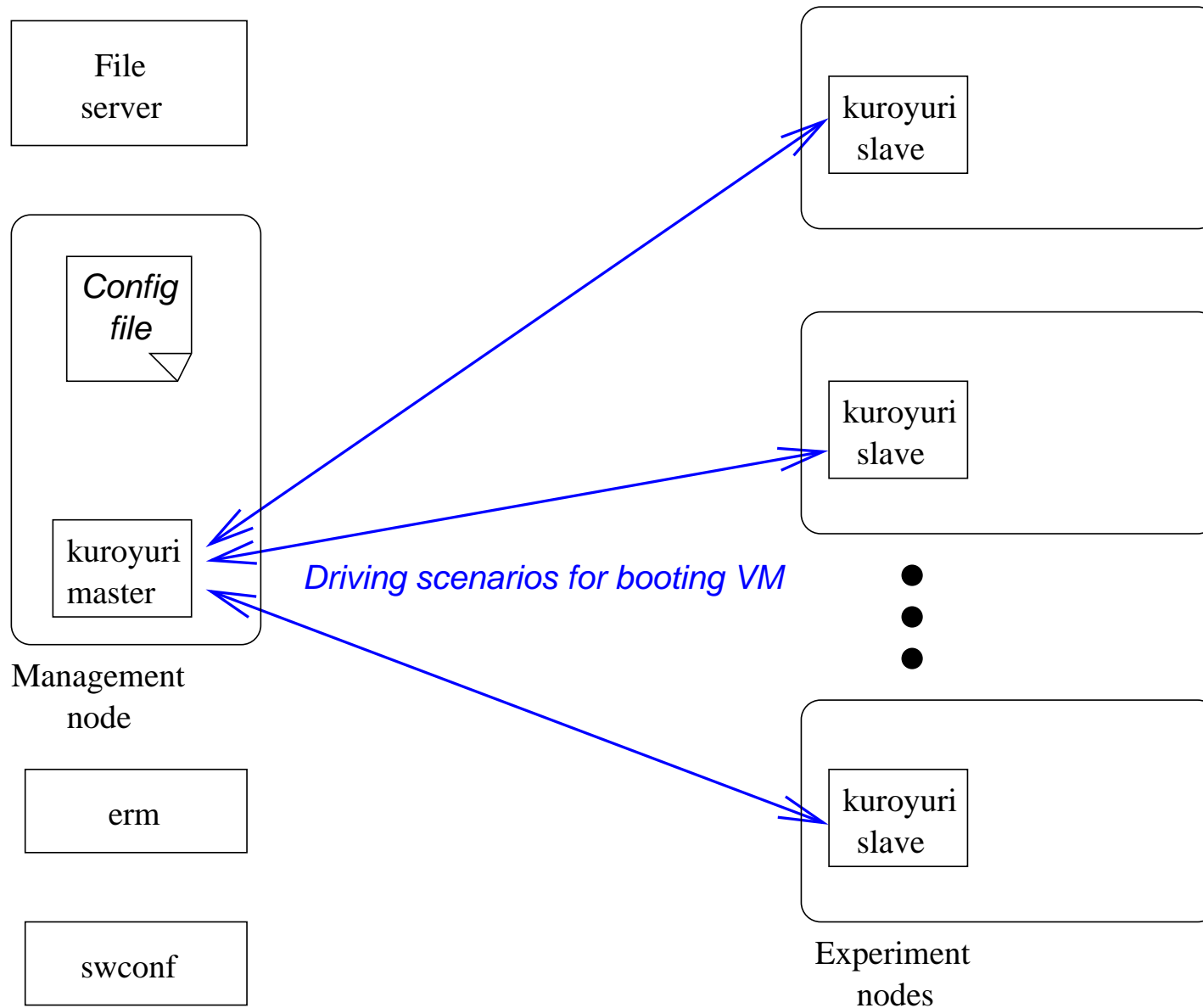


- ある実ノード上で起動できる仮想機械のエントリを登録
 - 実ノードで何台の仮想機械が起動できるかを設定
- VMware の MAC アドレスを決定
 - erm に物理ノードと同様に仮想機械のエントリを追加
 - 物理ノードの管理側 IP アドレスから連想できるものに

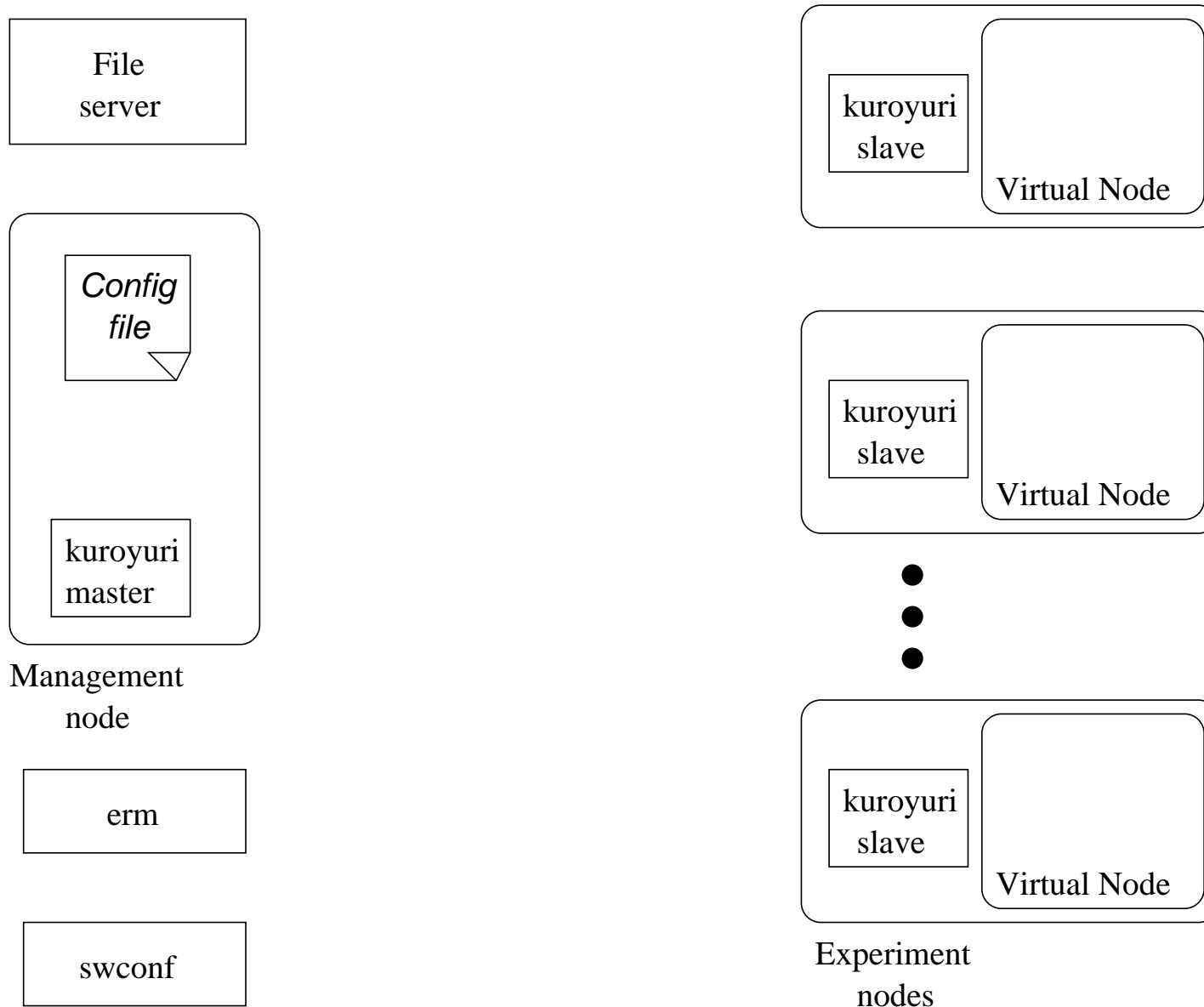
SpringOS/VMの処理手順



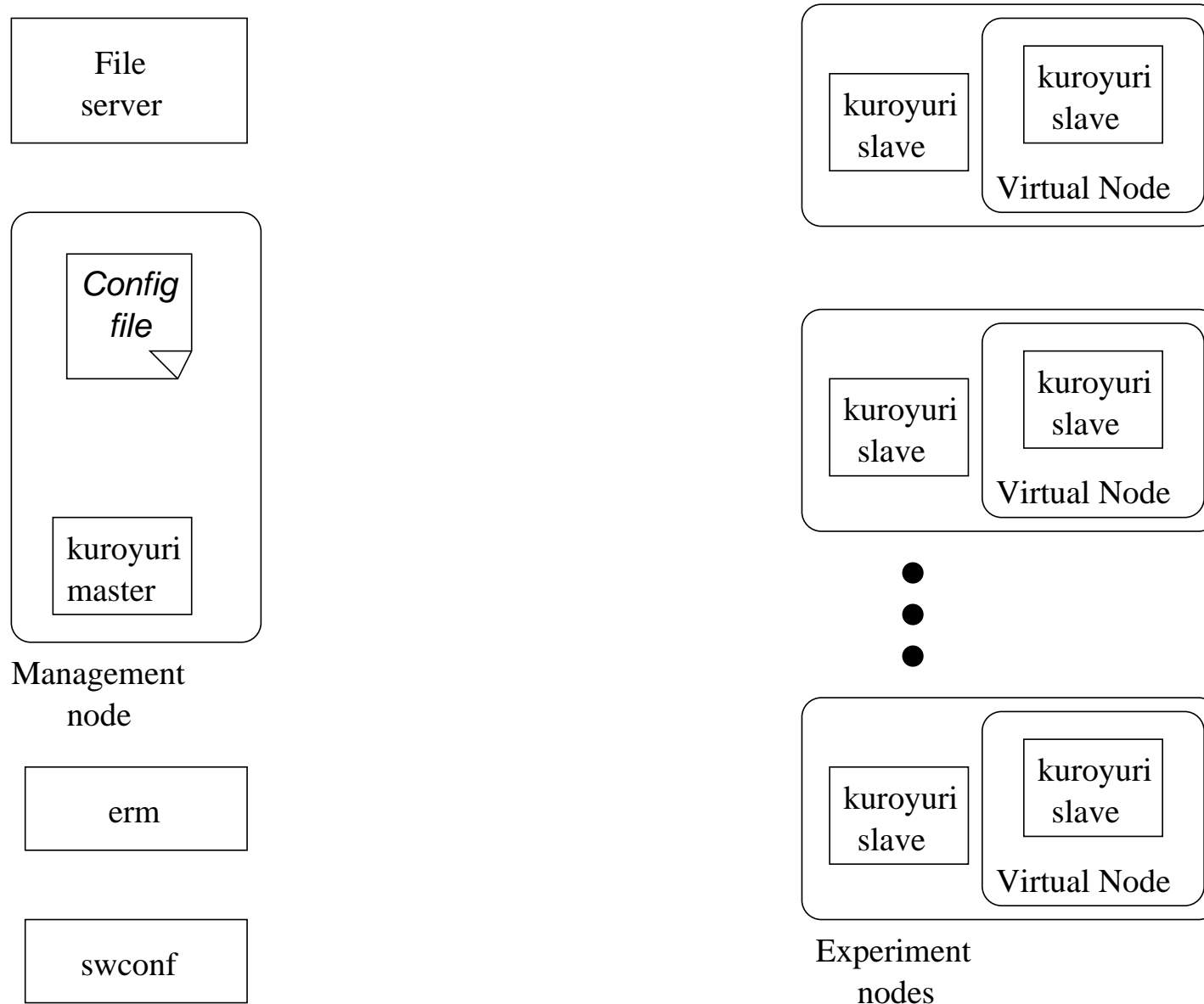
SpringOS/VMの処理手順



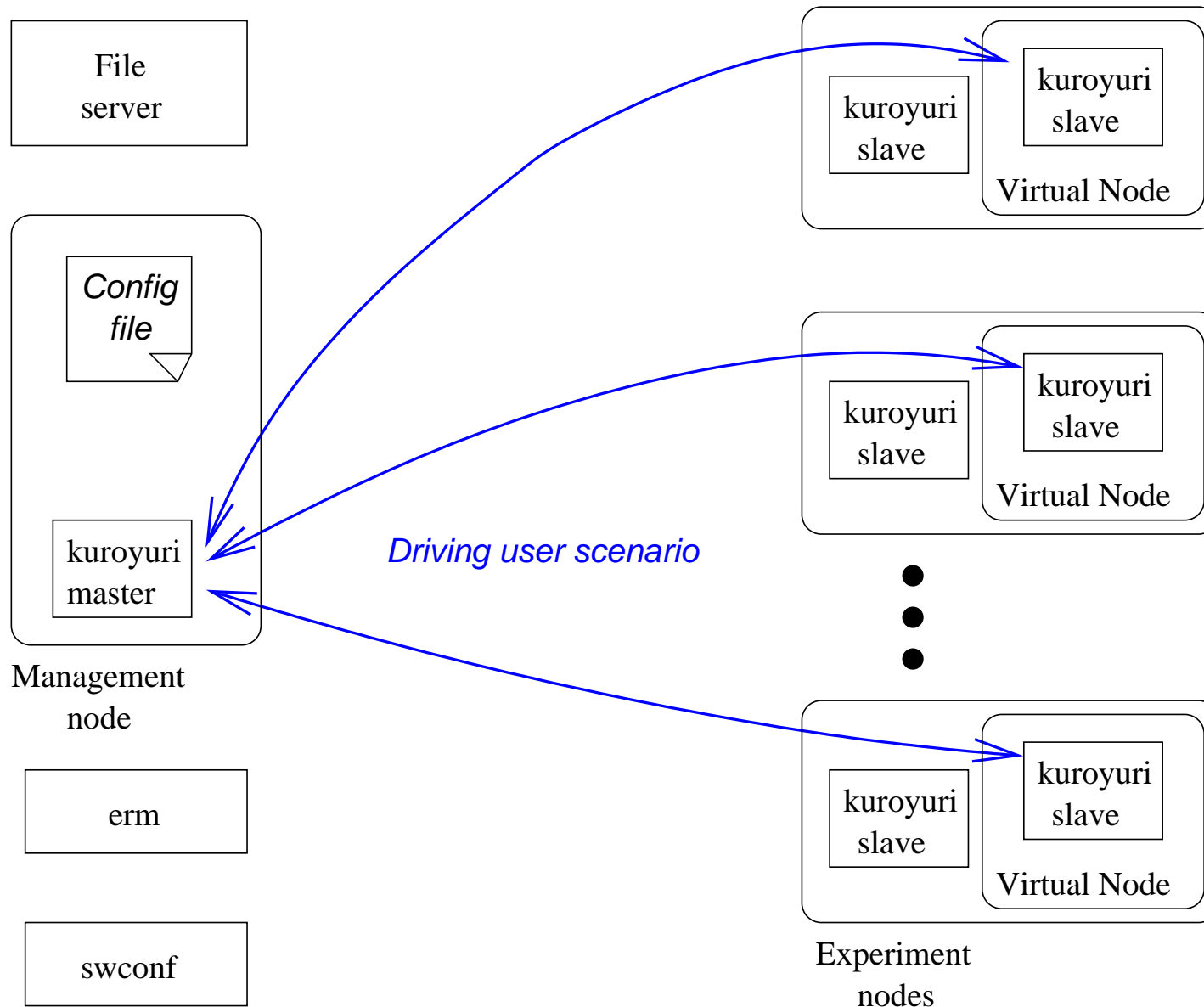
SpringOS/VMの処理手順



SpringOS/VMの処理手順



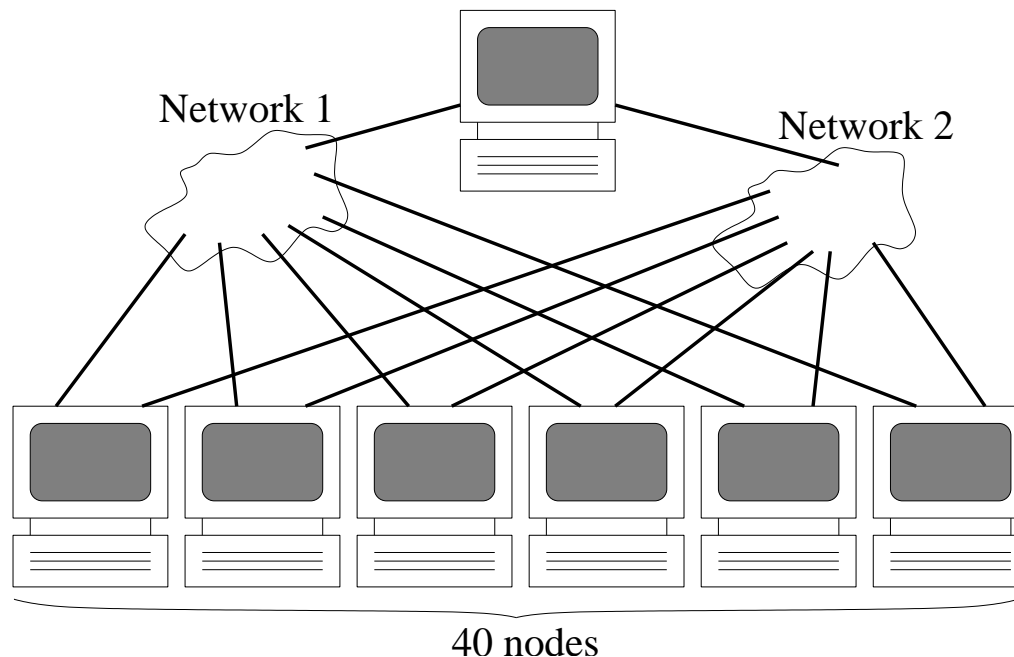
SpringOS/VMの処理手順



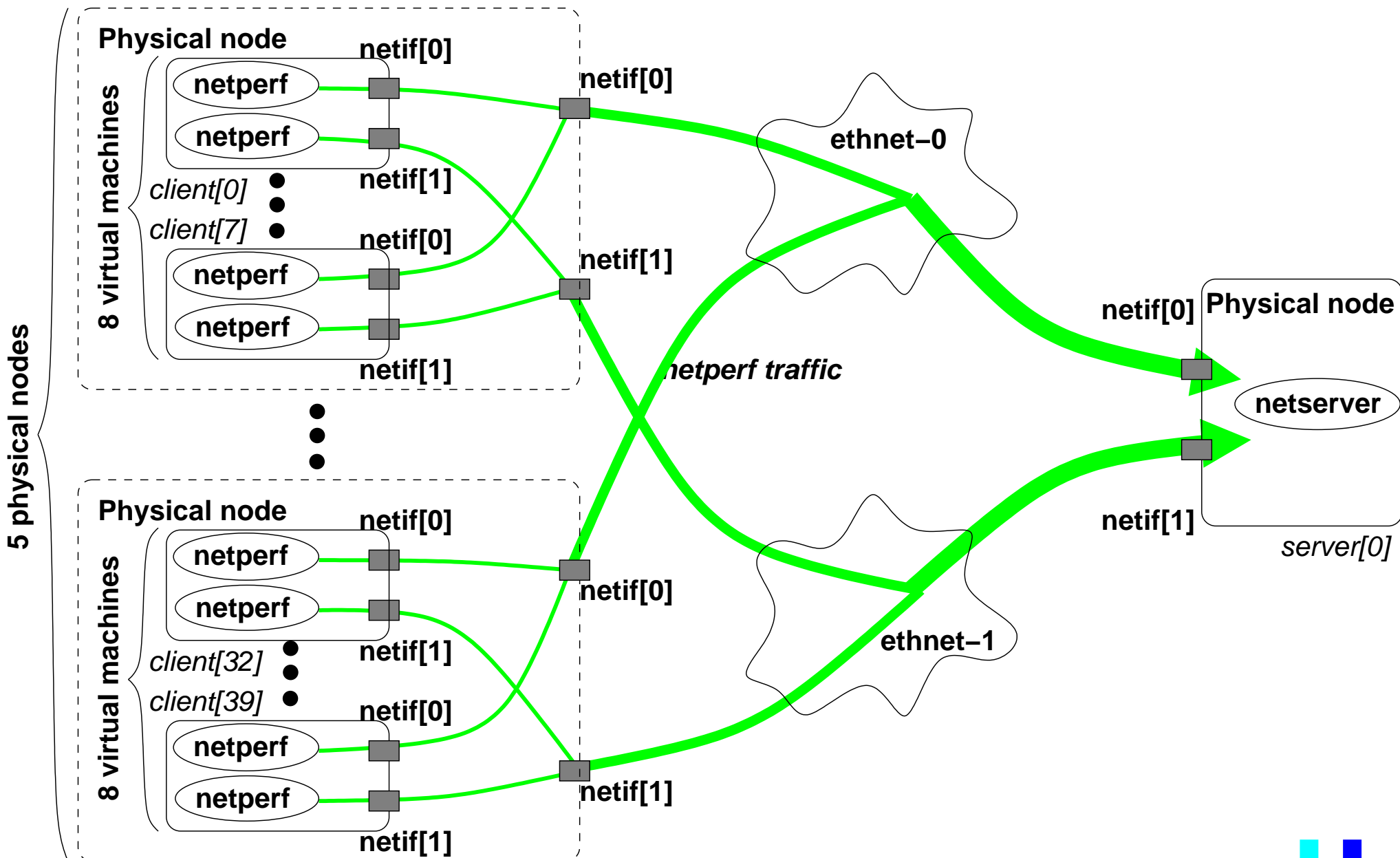
- vmnet-bridge を利用し VMware の I/F を物理ノードの I/F に関連付け
- VMware のディスクイメージのダウンロード
- VMware の設定ファイル変更
 - MAC アドレス
 - vmnet へ仮想機械の接続
- VMware の I/F が関連付けされている物理ノードの I/F が接続されているスイッチの VLAN 設定

実験例と設定例

- netperf での帯域測定
- ノード
 - 5 台を VMware で 8 多重 (合計 40 台)
 - 一台は物理ノードをそのまま利用
- StarBED で実施



実験トポロジ



```
nodeclass clC {
  maxvnodes 8
  vntype vmwareC
  ostype "Linux"
  diskimage "ftp://172.16.3.253/linux.vmdk"
  netif media fastethernet via 0 net "ethnet-0"
  netif media fastethernet via 1 net "ethnet-1"
  scenario {
    netifit "/usr/local/bin/ifsetup/src/ifscan"
    wakewait "/sbin/ifconfig" "/sbin/ifconfig" self.netif[0].rname\\
      (haddr(self.netif[0].ipaddr))
    recv dstA
    recv dstB
    wake "/usr/local/bin/netperf" "/usr/local/bin/netperf" "-H" dstA
    wakewait "/usr/local/bin/netperf" "/usr/local/bin/netperf" "-H" dstB
    send "cdone"
  }
}
nodeset client class clC num 40
nodeset server class svC num 1
```

➤ まとめ

- 実ノードによる実験環境の拡張は困難
- SpringOS を拡張し仮想機械を制御
- 簡単な実験で想定通りの挙動を確認

➤ 今後の課題

- 一台の物理ノード上で複数のクラスに属する仮想機械を起動
- VMware の snapshot 機能などへの対応
- その他の仮想ノード実現技術の特性についての考察と実験環境への投入





SpringOS を構成するモジュール群

要素	役割
erm	リソースの状態・属性管理 実験へのリソース割り当て
swconf	スイッチの設定
kuroyuri master	設定記述の認識 ノードへのソフトウェア導入指示 ノードへのシナリオ配布 シナリオ実行補助 他のモジュール制御
ni	ノードへのソフトウェア導入 (ディスクレス OS 上で動作)
kuroyuri slave	ノードでのシナリオ実行

1. kuroyuri master による実験遂行者の設定ファイルの読み込み
2. erm によるリソースの割り当て
3. ノードの起動とノード上での ni 起動
4. ni と kuroyuri master によるノードへの OS およびアプリケーションの導入
5. ノード再起動
6. kuroyuri master から swconf にスイッチの設定リクエスト送信
7. スイッチ設定による実験トポロジ構築
8. kuroyuri master と kuroyuri slave によるシナリオ実行

VMWare 設定・起動用シナリオ

```
nodeclass vmwareC {
  method "HDD"
  disktype "IDE"
  partition 4
  ostype "Linux"
  diskimage "ftp://172.16.3.253/vmhost.gz"
  netif media fastethernet
  netif media fastethernet
  scenario {
    netifit "/usr/local/bin/ifscan"
    wakewait "/sbin/ifconfig" "/sbin/ifconfig" self.netif[0].rname\\
      (haddr(self.netif[0].ipaddr))
    wakewait "/usr/bin/wget" "/usr/bin/wget" "-q" "ftp://172.16.3.253/vmsetup.pl"
    wakewait "/bin/mv" "/bin/mv" "vmsetup.pl" "/tmp/vmsetup.pl"
    wakewait "/usr/bin/perl" "/usr/bin/perl" "/tmp/vmsetup.pl" "/tmp/vm.hint"
    for(i=0;i<_launchvnodes;i++) {
      wake "/usr/bin/vmware" "/usr/bin/vmware" "-q" "-x"\\
        ("/root/vmware/linux0"+toString(i)+"/linux.cfg")
    }
  }
}
```

- **大規模ネットワーク実証環境**
 - 512 台の PC クラスタ
 - 実験用ネットワークと管理用ネットワークが分離
 - 物理的ネットワーク接続を変更せずに実験トポロジを仮想的に構築
 - JGNII と WIDE による 2 本の 10Gbps 接続
 - VMware を利用して 1000 ノード規模の実験実績あり
 - <http://www.starbed.org/>