Módulo Ping

O módulo **ping** fornece funções para testar conectividade de rede através de ICMP, TCP e DNS. Este módulo é útil para monitoramento de disponibilidade, diagnóstico de problemas de rede e medição de latência em ambientes de produção.

Características principais:

- Suporte a ICMP ping (IPv4 e IPv6)
- Testes de porta TCP
- Configuração flexível de parâmetros
- Resolução DNS automática
- Timeout configurável

Funções Disponíveis

1. `ping.up([host], [count], [data_size], [timeout], [interval_ms])`

Verifica se um host está acessível através de ping ICMP.

Parâmetros:

- **host** (opcional, string): Endereço do host ou nome de domínio (ex: "google.com", "192.168.1.1")
- **count** (opcional, número, padrão: `params.icmpUpNumPackets` ou 3): Número de tentativas de ping
- **data_size** (opcional, número, padrão: 24): Tamanho do pacote de dados em bytes
- **timeout** (opcional, número, padrão: `params.icmpUpTimeout` ou 2): Timeout em segundos para cada tentativa
- **interval_ms** (opcional, número, padrão: `params.icmpUpInterval` ou 100): Intervalo entre tentativas em milissegundos
- Se `host` não for fornecido, usa `params.address`
- Se `count` não for fornecido, usa `params.icmpUpNumPackets`
- Se `timeout` não for fornecido, usa `params.icmpUpTimeout`
- Se `interval_ms` não for fornecido, usa `params.icmpUpInterval`

Retorno:

- **boolean**: `true` se pelo menos um ping foi bem-sucedido, `false` caso contrário

Exemplo de Uso:

```
-- Verificar se um host está acessível
local acessivel = ping.up("google.com")
-- acessivel = true (se responder ao ping)

-- Verificar com parâmetros personalizados
local resultado = ping.up("servidor.local", 5, 32, 5, 200)
-- 5 tentativas, pacotes de 32 bytes, timeout de 5s, intervalo de 200ms

-- Testar endereço IPv6
local ipv6_ok = ping.up("2001:4860:4860::8888")
-- Testa conectividade IPv6 com Google DNS

-- Verificar múltiplos hosts
local hosts = {"router.local", "192.168.1.1", "8.8.8.8"}
for _, host in ipairs(hosts) do
    if ping.up(host) then
        print(host .. " está online")
    else
        print(host .. " está offline")
    end
end
```

2. `ping.send([host], [count], [data_size], [timeout], [interval_ms])`

Mede a latência de ping ICMP para um host.

Parâmetros:

- **host** (opcional,string): Endereço do host ou nome de domínio
- **count** (opcional, número, padrão: `params.icmpTimeNumPackets` ou 3): Número de tentativas de ping
- **data_size** (opcional, número, padrão: 24): Tamanho do pacote de dados em bytes

- **timeout** (opcional, número, padrão: `params.icmpTimeTimeout` ou 2): Timeout em segundos para cada tentativa
- **interval_ms** (opcional, número, padrão: `params.icmpTimeInterval` ou 100): Intervalo entre tentativas em milissegundos
- Se `host` não for fornecido, usa `params.address`
- Se `count` não for fornecido, usa `params.icmpTimeNumPackets`
- Se `timeout` não for fornecido, usa `params.icmpTimeTimeout`
- Se `interval_ms` não for fornecido, usa `params.icmpTimeInterval`

Retorno:

- **tuple**: `(latencia, erro)` onde:
  - **latencia** (número ou nil): Latência média em milissegundos, ou `nil` se falhar
  - **erro** (string ou nil): Mensagem de erro, ou `nil` se bem-sucedido

Exemplo de Uso:

```
-- Medir latência básica
local latencia, erro = ping.send("google.com")
-- latencia = 25.3 (exemplo), erro = nil

-- Medir com parâmetros personalizados
local latencia, erro = ping.send("servidor.remoto", 10, 64, 5, 500)
-- 10 tentativas, pacotes de 64 bytes, timeout 5s, intervalo 500ms

-- Tratar resultado
local latencia, erro = ping.send("host.inacessivel")
if latencia then
    print("Latência: " .. latencia .. "ms")
else
    print("Erro: " .. erro)
    -- erro = "ping failed" ou mensagem específica
end

-- Comparar latência entre múltiplos hosts
local hosts = {
    "dns.google",
    "cloudflare-dns.com",
    "quad9.net"
}
```

```
for _, host in ipairs(hosts) do
    local latencia, erro = ping.send(host, 5)
    if latencia then
        print(host .. ": " .. string.format("%.2f", latencia) .. "ms")
    end
end
```

3. `ping.port(host, port, [count], [timeout])`

Testa conectividade a uma porta TCP específica.

Parâmetros:

- **host** (string): Endereço do host ou nome de domínio
- **port** (número): Número da porta TCP a testar
- **count** (opcional, número, padrão: 3): Número de tentativas de conexão
- **timeout** (opcional, número, padrão: 2): Timeout em segundos para cada tentativa

Retorno:

- **tuple**: `(latencia, erro)` onde:
  - **latencia** (número ou nil): Latência média de conexão em milissegundos, ou `nil` se falhar
  - **erro** (string ou nil): Mensagem de erro, ou `nil` se bem-sucedido

Exemplo de Uso:

```
-- Testar porta HTTP padrão
local latencia, erro = ping.port("google.com", 80)
-- latencia = 45.2 (exemplo), erro = nil

-- Testar porta HTTPS
local latencia, erro = ping.port("google.com", 443)
if latencia then
    print("HTTPS acessível com latência: " .. latencia .. "ms")
else
    print("HTTPS inacessível: " .. erro)
end
```

```
-- Testar múltiplas portas
local portas = {22, 80, 443, 3306, 5432}
for _, porta in ipairs(portas) do
    local latencia, erro = ping.port("servidor.local", porta, 2, 3)
    if latencia then
        print("Porta " .. porta .. " aberta (" .. latencia .. "ms)")
    else
        print("Porta " .. porta .. " fechada ou inacessível")
    end
end

-- Verificar serviço específico
local function verificar_servico(host, porta, nome_servico)
    local latencia, erro = ping.port(host, porta)
    if latencia then
        log.info(nome_servico .. " OK (" .. latencia .. "ms)")
        return true
    else
        log.error(nome_servico .. " FALHA: " .. erro)
        return false
    end
end
```

Informações Adicionais

Resolução DNS Automática

Todas as funções realizam resolução DNS automaticamente:

```
-- Funciona com nomes de domínio
ping.up("google.com")
ping.send("api.github.com")
ping.port("servidor.local", 443)

-- Funciona com endereços IP
ping.up("8.8.8.8")
ping.send("192.168.1.1")
```

```
ping.port("10.0.0.5", 22)
```

Suporte a IPv4 e IPv6

O módulo detecta automaticamente o tipo de endereço:

```
-- IPv4
ping.up("8.8.8.8")

-- IPv6
ping.up("2001:4860:4860::8888")

-- Nome de domínio (resolvido para IPv4 ou IPv6)
ping.up("google.com")
```

Timeout Configurável

Cada função permite configurar timeout individual:

```
-- Timeout curto para redes locais
ping.up("router.local", 3, 24, 1)  -- 1 segundo

-- Timeout longo para redes com alta latência
ping.up("servidor.remoto", 3, 24, 10)  -- 10 segundos

-- Timeout específico por porta
ping.port("database.remoto", 5432, 3, 5)  -- 5 segundos
```

Revision #1
Created 3 February 2026 16:44:03 by Marc
Updated 3 February 2026 16:44:16 by Marc