Internet-Monitoring auf RPI

Prüft auf

- Router Verfügbarkeit
- Third Hop beim Aufrufen von Google DNS (Verfügbarkeit des Services vom Provider)
- Verfügbarkeit von DNS-Auflösung
- Möglichkeit Daten per HTTP zu bekommen

Achtet auf Installationen von PiHole oder Adguard Home, die die Prüfmethode für den Provider-Hop nach traceroute brechen würden...

Log Rotation bei 10 MB.

Raspbian bringt traceroute nicht von Haus aus mit

```
sudo apt install traceroute
```

Generator Claude Sonnet 4 für Linux 6.6.28+rpt-rpi-v8 aarch64

```
#!/bin/bash

# Internet Connection Monitor Script für Raspbian
# Überwacht Gateway-Erreichbarkeit und Internet-Konnektivität
# Autor: Generated Script
# Version: 1.0

# Konfiguration
LOG_FILE="/var/log/internet_monitor.log"
MAX_LOG_SIZE=10485760  # 10MB in Bytes
TIMEOUT=5              # Ping Timeout in Sekunden
PING_COUNT=3           # Anzahl Ping-Pakete

# DNS Server für Konnektivitätstests
DNS_SERVERS=("8.8.8.8" "1.1.1.1" "9.9.9.9")

# Farben für Terminal-Ausgabe (falls direkt ausgeführt)
RED='\033[0;31m'
GREEN='\033[0;32m'
```

```
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

# Funktion: Timestamp generieren
get_timestamp() {
    date '+%Y-%m-%d %H:%M:%S'
}

# Funktion: Log-Datei rotieren wenn zu groß
rotate_log() {
    if [[ -f "$LOG_FILE" && $(stat -f%z "$LOG_FILE" 2>/dev/null || stat -c%s "$LOG_FILE"
2>/dev/null) -gt $MAX_LOG_SIZE ]]; then
        mv "$LOG_FILE" "${LOG_FILE}.old"
        echo "$(get_timestamp) - Log-Datei rotiert" >> "$LOG_FILE"
    fi
}

# Funktion: Gateway ermitteln
get_default_gateway() {
    ip route show default | awk '/default/ { print $3; exit }'
}

# Funktion: Provider Next Hop ermitteln (echter Provider-Hop, nicht AdGuard)
get_provider_nexthop() {
    local gateway=$(get_default_gateway)
    if [[ -n "$gateway" ]]; then
        # Traceroute zum ersten externen DNS-Server
        local traceroute_output=$(traceroute -n -m 5 -w 2 8.8.8.8 2>/dev/null)

        # Durchsuche Hops 2-4 nach dem ersten nicht-privaten IP (Provider-Hop)
        for hop in {2..4}; do
            local hop_ip=$(echo "$traceroute_output" | awk -v hop="$hop" 'NR==hop+1 {print
$2}' | head -1)

            # Prüfe ob IP gesetzt und nicht leer/privat/loopback ist
            if [[ -n "$hop_ip" && "$hop_ip" != "*" && "$hop_ip" != "* * *" ]]; then
                # Prüfe ob es eine öffentliche IP ist (nicht RFC1918/Loopback/AdGuard)
                if ! is_private_ip "$hop_ip"; then
                    echo "$hop_ip"
```

```
                    return 0
                fi
            fi
        done
    fi
    echo ""
}

# Funktion: Prüfen ob IP privat/loopback/AdGuard ist
is_private_ip() {
    local ip="$1"

    # Prüfe auf private/reservierte IP-Bereiche
    if [[ "$ip" =~ ^10\. ]] || \
       [[ "$ip" =~ ^172\.(1[6-9]|2[0-9]|3[01])\. ]] || \
       [[ "$ip" =~ ^192\.168\. ]] || \
       [[ "$ip" =~ ^127\. ]] || \
       [[ "$ip" =~ ^192\.0\.0\. ]] || \
       [[ "$ip" =~ ^169\.254\. ]]; then
        return 0  # ist privat
    else
        return 1  # ist öffentlich
    fi
}

# Funktion: Netzwerk-Interface des Standard-Gateways ermitteln
get_gateway_interface() {
    ip route show default | awk '/default/ { print $5; exit }'
}

# Funktion: Ping-Test durchführen
ping_test() {
    local target="$1"
    local description="$2"

    if ping -c $PING_COUNT -W $TIMEOUT "$target" >/dev/null 2>&1; then
        echo "$(get_timestamp) - ✓ $description ($target) - ERREICHBAR" >> "$LOG_FILE"
        return 0
    else
```

```
        echo "$(get_timestamp) - ✗ $description ($target) - NICHT ERREICHBAR" >> "$LOG_FILE"
        return 1
    fi
}

# Funktion: DNS-Auflösung testen
dns_test() {
    local test_domain="google.com"

    if nslookup "$test_domain" >/dev/null 2>&1; then
        echo "$(get_timestamp) - ✓ DNS-Auflösung ($test_domain) - FUNKTIONIERT" >> "$LOG_FILE"
        return 0
    else
        echo "$(get_timestamp) - ✗ DNS-Auflösung ($test_domain) - FEHLGESCHLAGEN" >>
"$LOG_FILE"
        return 1
    fi
}

# Funktion: HTTP/HTTPS Konnektivität testen
http_test() {
    local test_url="http://www.google.com"

    if curl -s --connect-timeout $TIMEOUT --max-time $((TIMEOUT*2)) "$test_url" >/dev/null
2>&1; then
        echo "$(get_timestamp) - ✓ HTTP-Konnektivität ($test_url) - FUNKTIONIERT" >>
"$LOG_FILE"
        return 0
    else
        echo "$(get_timestamp) - ✗ HTTP-Konnektivität ($test_url) - FEHLGESCHLAGEN" >>
"$LOG_FILE"
        return 1
    fi
}

# Funktion: Netzwerk-Informationen sammeln
collect_network_info() {
    local gateway=$(get_default_gateway)
    local provider_hop=$(get_provider_nexthop)
```

```
    local interface=$(get_gateway_interface)
    local ip_addr=$(ip addr show "$interface" 2>/dev/null | grep "inet " | head -n1 | awk
'{print $2}' | cut -d'/' -f1)

    echo "$(get_timestamp) - Netzwerk-Info: Interface=$interface, IP=$ip_addr,
Gateway=$gateway, Provider-NextHop=$provider_hop" >> "$LOG_FILE"
}

# Funktion: Vollständige Verbindungsprüfung
check_connectivity() {
    echo "$(get_timestamp) - === Internet-Konnektivitätsprüfung gestartet ===" >> "$LOG_FILE"

    # Log rotieren falls nötig
    rotate_log

    # Netzwerk-Informationen sammeln
    collect_network_info

    local gateway=$(get_default_gateway)
    local provider_nexthop=$(get_provider_nexthop)
    local gateway_reachable=false
    local provider_reachable=false
    local internet_reachable=false
    local dns_working=false
    local http_working=false

    # Gateway-Erreichbarkeit prüfen
    if [[ -n "$gateway" ]]; then
        if ping_test "$gateway" "Gateway"; then
            gateway_reachable=true
        fi
    else
        echo "$(get_timestamp) - ✗ Kein Standard-Gateway gefunden!" >> "$LOG_FILE"
    fi

    # Provider Next Hop prüfen (wichtigster Test für Providerseite!)
    if $gateway_reachable; then
        if [[ -n "$provider_nexthop" ]]; then
            if ping_test "$provider_nexthop" "Provider Next Hop"; then
```

```
                provider_reachable=true
            fi
        else
            echo "$(get_timestamp) - ! Provider Next Hop konnte nicht ermittelt werden
(AdGuard/Firewall?)" >> "$LOG_FILE"
            # Fallback: Teste externe DNS direkt
            for dns_server in "${DNS_SERVERS[@]}"; do
                if ping_test "$dns_server" "Fallback Provider Test"; then
                    provider_reachable=true
                    break
                fi
            done
        fi
    fi

    # DNS-Server-Erreichbarkeit prüfen (falls Provider erreichbar)
    if $provider_reachable || ($gateway_reachable && [[ -z "$provider_nexthop" ]]); then
        for dns_server in "${DNS_SERVERS[@]}"; do
            if ping_test "$dns_server" "DNS-Server"; then
                internet_reachable=true
                break
            fi
        done
    fi

    # DNS-Auflösung testen
    if $internet_reachable; then
        if dns_test; then
            dns_working=true
        fi
    fi

    # HTTP-Konnektivität testen
    if $dns_working; then
        if http_test; then
            http_working=true
        fi
    fi
```

```
    # Gesamtstatus bestimmen
    local status="UNBEKANNT"
    local status_code=3

    if $http_working; then
        status="VOLLSTÄNDIG FUNKTIONSFÄHIG"
        status_code=0
    elif $dns_working; then
        status="DNS FUNKTIONIERT, HTTP PROBLEME"
        status_code=1
    elif $internet_reachable; then
        status="INTERNET ERREICHBAR, DNS PROBLEME"
        status_code=1
    elif $provider_reachable; then
        status="PROVIDER ERREICHBAR, INTERNET PROBLEME"
        status_code=2
    elif $gateway_reachable; then
        status="NUR LOKALES NETZWERK - PROVIDER NICHT ERREICHBAR"
        status_code=2
    else
        status="KEINE NETZWERKVERBINDUNG"
        status_code=3
    fi

    echo "$(get_timestamp) - STATUS: $status (Code: $status_code)" >> "$LOG_FILE"
    echo "$(get_timestamp) - === Prüfung beendet ===" >> "$LOG_FILE"
    echo "" >> "$LOG_FILE"

    # Status auch auf stdout ausgeben (für manuelle Ausführung)
    if [[ -t 1 ]]; then
        case $status_code in
            0) echo -e "${GREEN}✓ Internet: $status${NC}" ;;
            1) echo -e "${YELLOW}⚠ Internet: $status${NC}" ;;
            2) echo -e "${YELLOW}⚠ Internet: $status${NC}" ;;
            3) echo -e "${RED}✗ Internet: $status${NC}" ;;
        esac
    fi

    return $status_code
```

```
}

# Funktion: Hilfe anzeigen
show_help() {
    echo "Internet Connection Monitor Script"
    echo ""
    echo "Verwendung: $0 [OPTION]"
    echo ""
    echo "Optionen:"
    echo "  -h, --help     Zeige diese Hilfe"
    echo "  -l, --log      Zeige die letzten Log-Einträge"
    echo "  -s, --status   Führe eine einmalige Statusprüfung durch"
    echo "  -c, --config   Zeige aktuelle Konfiguration"
    echo ""
    echo "Standard-Verhalten: Führe Konnektivitätsprüfung durch"
    echo ""
    echo "Log-Datei: $LOG_FILE"
}

# Funktion: Log-Einträge anzeigen
show_log() {
    local lines=${1:-50}
    if [[ -f "$LOG_FILE" ]]; then
        echo "Letzte $lines Zeilen aus $LOG_FILE:"
        echo "=" $(printf '=%.0s' {1..60})
        tail -n "$lines" "$LOG_FILE"
    else
        echo "Log-Datei $LOG_FILE nicht gefunden."
    fi
}

# Funktion: Konfiguration anzeigen
show_config() {
    echo "Aktuelle Konfiguration:"
    echo "======================="
    echo "Log-Datei: $LOG_FILE"
    echo "Max. Log-Größe: $((MAX_LOG_SIZE/1024/1024)) MB"
    echo "Ping-Timeout: $TIMEOUT Sekunden"
    echo "Ping-Count: $PING_COUNT Pakete"
```

```
    echo "DNS-Server: ${DNS_SERVERS[*]}"
    echo ""
    echo "Aktuelles Gateway: $(get_default_gateway)"
    echo "Gateway-Interface: $(get_gateway_interface)"
    echo "Provider Next Hop: $(get_provider_nexthop)"
}

# Hauptprogramm
main() {
    # Root-Rechte prüfen (für Log-Datei in /var/log)
    if [[ $EUID -ne 0 && "$LOG_FILE" == /var/log/* ]]; then
        echo "Warnung: Root-Rechte benötigt für Log-Datei in /var/log/"
        echo "Verwende temporäre Log-Datei: /tmp/internet_monitor.log"
        LOG_FILE="/tmp/internet_monitor.log"
    fi

    # Parameter verarbeiten
    case "${1:-}" in
        -h|--help)
            show_help
            exit 0
            ;;
        -l|--log)
            show_log "${2:-50}"
            exit 0
            ;;
        -s|--status)
            check_connectivity
            exit $?
            ;;
        -c|--config)
            show_config
            exit 0
            ;;
        "")
            check_connectivity
            exit $?
            ;;
        *)
```

```
            echo "Unbekannte Option: $1"
            echo "Verwende '$0 --help' für Hilfe."
            exit 1
            ;;
    esac
}

# Script ausführen
main "$@"
```

Installation

```
sudo cp internet_monitor.sh /usr/local/bin/
sudo chmod +x /usr/local/bin/internet_monitor.sh
```

Test

```
sudo /usr/local/bin/internet_monitor.sh --status
```

Cronjob

```
*/5 * * * * /usr/local/bin/internet_monitor.sh >/dev/null 2>&1
```

Version #5
Erstellt: 2025-08-24 12:20:29 UTC von Konstantin
Zuletzt aktualisiert: 2025-08-24 12:42:13 UTC von Konstantin