Formát souboru relay.conf

Příklad

```
# Tato radka znamena komentar.
; Tohle je taky komentar.
source port ttyS0,9600,8N1,RTU,NOFLOW
 id 6 => host 192.168.2.120 frame_t 500ms pend_t 2s tx_t 3s id 0
 id 4 => host 192.168.2.120 id 1
 id 5 => host 192.168.2.121 id 2
 id 7 => host senzor22.in.firma.cz:540 id 0
 id 8 => host senzor22.in.firma.cz:540 id 1
# Prazdne radky jsou ignorovany.

source port ttyS1,9600,8N1,ASCII,FLOW frame_t 5ch pend_t 500ms tx_t 200000
 id 6 => host 192.168.2.120 id 1
 id 4 => host 192.168.2.120 id 3
 id * => port ttyS2,19200,8N1,RTU,FLOW gw_timeout

source host any:503 gw_nopath
 id 0xE   => host nekdo.nekde.cz id 50
 id 0x0F  => port ttyS2
```

Formálnější definice syntaxe

Na samostatném řádku je určen zdroj dat (Modbus port typu slave).

Na následujících řádcích jsou určena pravidla pro mapování Modbus Device ID's, tj. mj. cílové porty. Jeden řádek = jedno pravidlo = jeden cíl (Modbus port typu master).

Pravidla přísluší k naposledy definovanému zdroji. Definice cíle se mohou opakovat (duplicita je detekována a ošetřena).

Z logiky věci vyplývá, že prvním výkonným řádkem v konfiguračním souboru musí být definice zdroje dat.

Duplicitní zdroje nejsou povoleny.

Duplicitní cíle jsou povoleny, interně jsou všechny duplicitní cíle redukovány na jediný cíl, který si podrží vedlejší atributy z prvotní definice (vedlejší atributy udané v dalších definicích jsou potichu ignorovány).

Není povoleno definovat cíl, který byl již definován jako zdroj – jeden port nemůže být zároveň master a slave, ani zároveň lokální Modbus/TCP server a cílový port (vznikla by smyčka).

Jak zdroj tak cíl může být typu TCP, RTU nebo ASCII.

Formát zdroje:

```
source <TCP host | serial port> [options]
```

Formát pravidla a cíle:

```
id <src id | *> => <TCP host | serial port> [options] [id <dst id>]
```

Pomocí zástupného znaku * (hvězdička) lze zadat obecné či nespecifické pravidlo, které se uplatní na všechna ID v příchozích rámcích – relay bude propouštět všechny dotazy. Uplatňují se přednostně specifická pravidla - teprve pokud není nalezeno žádné specifické pravidlo, uplatní se pravidlo obecné.

Pokud nezadáme cílové ID pravidla (prostě vynecháme celý člen včetně klíčového slova „id“), zůstaně ID nezměněno – nebude se překládat.

Formát definice sériového portu:

```
port <device node>[,option[,option[,option]]]
options:
 - baud rate (default: 9600)
 - Databits Parity Stopbits (default: 8N1)
 - FLOW | NOFLOW (default: NOFLOW)
 - RTU | ASCII (default: RTU)
```

Device node je jméno speciálního souboru, pod kterým se v Unixu skrývá systémové zařízení. Nejlépe včetně cesty, jména v adresáři /dev i bez cesty. Například /dev/ttyS0 je totéž jako COM1, /dev/ttyS1 je totéž jako COM2 apod. (sériová „konzolová“ zařízení v Unixu jsou číslována od nuly).

Jako rychlost portu je možno vybrat z obecně známých typizovaných rychlostí, které jsou charakteristické pro UART 16550A, přesněji které jsou vyjmenovány v hlavičkových souborech systémové knihovny v Linuxu – od 50 bd po 230400 bd.

Datových bitů může být 5-8, parita O | E | N, stop bity 1 nebo 2.

Flow control je buď hardwarová (RTS+CTS) nebo žádná.

Formát definice TCP hostitele:

```
host <DNS name | IP Address>[:TCP_port]
```

Nepovinné volby, společné pro sériové i TCP porty:

1) timeouty

```
frame_t <timeout>     = frame break timeout
pend_t  <timeout>     = pending queue timeout (master ports only)
tx_t    <timeout>     = transmit queue timeout
```

2) volby povolující hlášení gatewayových chyb směrem k masteru

```
gw_nopath     = Modbus gateway reports „no path defined for this ID“
gw_timeout    = Modbus gateway reports „no response from the slave“
```

Všechny druhy timeoutů se udávají ve formátu <číslo>[jednotka] bez mezery, např. 10000, 10ms nebo 9ch. Pokud chybí jednotka, použijí se implicitní mikrosekundy – alternativně lze specifikovat jednotku „ms“, „s“ nebo „ch“. Posledně jmenovaná znamená „characters“, tj. násobek ekvivalentu doby odeslání jednoho znaku na dané jmenovité rychlosti sériového portu (počítáno jako 10 bitů). Tato jednotka se hodí především u sběrnic typu Modbus/RTU – tento standard specifikuje konec rámce jako pauzu v délce nejméně 3.5 znaku.

Volba `frame_t` (frame break timeout) znamená maximální povolenou mezeru mezi dvěma znaky, které mají ještě patřit do jednoho rámce. Default pro tento timeout je závislý na typu sběrnice:

RTU = 4 * ekvivalent 1 znaku (tj. na rychlosti 9600 bps cca 4-5 ms), norma říká 3.5ch
ASCII = 30 sekund (norma říká, že rámce se rozhodně nemají ukončovat podle timeoutu)
TCP = 100 ms (norma nestanoví konec rámce na základě timeoutu, ale v praxi běžně přijde jeden Modbus rámec zabalený v jediném TCP rámci, takže nízký timeout dává smysl)

Volba `pend_t` (pending queue timeout) znamená, jak dlouho se má čekat na odpověď po odeslání transakce (resp. po vstupu do syscallu write()) na master portu. Na slave portech „pending“ fronta vůbec není, takže u nich tento timeout nedává smysl. Default je 2 s – takto dlouhý timeout se uplatní na pomalých linkách při TCP komunikaci, takto dlouhá odezva od koncového zařízení je méně pravděpodobná.

Volba `tx_t` (transmit queue timeout) znamená, po jaké době se mají neodeslané transakce čistit z odesílací fronty konkrétního portu (master nebo slave). Z fronty odebírá transakce přirozeným způsobem odesílající thread, který volá write() a na master portech přeřazuje transakce do fronty „právě zpracovávaných transakcí“ (pending queue). Jinak řečeno, pokud je vysílání z nějakého důvodu pomalejší než zařazování dalších transakcí, třeba proto, že se nevracejí odpovědi na odeslané transakce, může se stát, že transakce již zařazené k odeslání vytimeoutují dříve, než se jich ujme odesílací vlákno knihovny. Default je 500 ms pro všechny druhy portů.

Pokud se týče signalizace chyb vzniklých na vzdálených sběrnicích, démon relay standardně žádné chybové odpovědi sám negeneruje – pouze transparentně předává standardní chybové odpovědi koncových zařízení, neboť se jedná o standardní Modbusové rámce.

Standard Modbus nicméně specifikuje dva druhy chybových odpovědí, které zde připadají v úvahu:

1) **Gateway path unavailable**
   Gateway neví, kam má dále poslat přijatý dotaz. V našem případě relay nemá definováno pravidlo, které by mohl uplatnit na zdrojové Modbus Unit ID. Pokud se tento problém vyskytne, jedná se patrně o chybu v konfiguraci celého systému (master, relay, slave).
2) **Gateway target device failed to respond**
   Od koncového zařízení nepřišla odpověď (relay zaznamenal timeout transakce). Koncové zařízení (slave) je patrně mrtvé, nebo neexistuje (chyba konfigurace).

Konfigurační soubor `relay.conf` umožňuje povolit hlášení těchto chybových stavů.

První hlášku povoluje volba `gw_nopath` . Tato chybová hláška pomůže odhalit případnou chybu v konfiguraci relaye (pokud nepoužíváme obecná pravidla).

Druhou hlášku povoluje volba `gw_timeout` . Tato chybová hláška pomůže rozlišit chyby před relayem a za relayem. Také může částečně zrychlit provoz v rozsáhlejším Modbusovém systému

s nestejně rychlými linkami a cyklickým pollingem, pokud se v něm občas vyskytují mrtvá koncová zařízení. Stačí na nejnižších relayích nastavit rozumně krátký `pend_t` plus ještě `gw_timeout` . Díky tomu se master dozví v nejkratší možné době, že vzdálený slave je nedostupný, a nemusí čekat na svůj dlouhý timeout, do kterého se musí vejít všechna koncová zařízení.

Poznámka ohledně časové přesnosti

Všechny timeouty jsou typicky lehce překračovány směrem nahoru, nikdy dolů. To vyplývá z chování použitých časovacích funkcí operačního systému.

Kvůli způsobu implementace použité modbusové knihovny (non-realtime user-space knihovna, nikoli např. kernel-space ovladač) jsou pro časování použity obyčejné user-space časovače. Proto přesnost „frame break timeoutu“ je v řádu cca jednotek ms (bez velké zátěže), pod větší zátěží se timeout protahuje až na cca 10 ms, což je perioda preemptivního systémového plánovače.

Aplikace i knihovna jsou napsány pokud možno důsledně tak, aby se vlákna navzájem zbytečně neblokovala a především aby zbytečně nespotřebovávala procesorový čas v nekonečných smyčkách. Bez výraznější zátěže CPU plánovač přepíná úlohy prakticky při každém blokujícím systémovém volání a při každém použití meziprocesové synchronizace.

Proto má Relay za normálních okolností přijatelné průchozí zpoždění – u korektních transakcí může zvyšovat latenci snad jen „frame break timeout“ v RTU, který ale pod přiměřenou zátěží není významně překračován.

Frontové timeouty jsou o něco méně přesné – mazání z fronty není zařízeno pomocí „řetězených časovačů“, ale pomocí časových značek na transakcích a periodicky spouštěného „garbage collectoru“ (samostatný thread na každém portu, který se pravidelně probouzí – součást knihovny). Perioda garbage collectoru je standardně 200 ms na master portech a 500 ms na slave portech – jedná se o hodnotu, která se zadává při kompilaci knihovny.

Doporučujeme vyvarovat se dotazů na slave zařízení, o kterých předem víme, že jsou mrtvá. Mrtvý slave brzdí svou sběrnici.
Toto je zřejmě problém hlavně u master portů na sériových sběrnicích s více slave zařízeními. Typický Modbus/TCP slave je na své TCP „sběrnici“ jediným koncovým zařízením, takže zablokuje pouze svůj vlastní port v aplikaci Relay a pak ještě dotazujícího se mastera – dotazy od jiných masterů na ostatní zařízení běží bez problémů dál.