Simulare il funzionamento di uno switch in Go

Nella gestione delle reti di computer, gli switch giocano un ruolo cruciale. A differenza degli hub, che inviano pacchetti a tutti i dispositivi sulla rete, gli switch sono in grado di inviare pacchetti solo al dispositivo di destinazione specifico. Questo miglioramento dell'efficienza è reso possibile grazie all'uso della tabella CAM (Content Addressable Memory), che mappa gli indirizzi MAC (Media Access Control) alle porte dello switch.

Gli switch operano principalmente al livello 2 del modello OSI (Data Link Layer). Il loro compito è di ricevere, processare e inoltrare i pacchetti di dati tra i dispositivi collegati. Gli switch mantengono una tabella CAM, che memorizza gli indirizzi MAC dei dispositivi collegati e le porte fisiche a cui sono associati. Quando un pacchetto arriva a una porta dello switch, l'indirizzo MAC di origine viene registrato nella tabella CAM insieme alla porta di ingresso. Se l'indirizzo MAC di destinazione è già noto, lo switch invia il pacchetto direttamente alla porta associata. Se non è noto, lo switch esegue una flooding, inviando il pacchetto a tutte le porte tranne quella di ingresso.

Per comprendere meglio il funzionamento di uno switch di rete, possiamo creare una semplice simulazione in Go. Questo esempio mostrerà come uno switch utilizza la tabella CAM per apprendere e inoltrare i pacchetti in base agli indirizzi MAC.

La struttura dello switch contiene una mappa che rappresenta la tabella CAM. La chiave della mappa è un indirizzo MAC e il valore è il numero della porta.

```
type Switch struct {
```

```
    camTable map[string]int // Tabella di
indirizzamento CAM
}
```

Il metodo `LearnMacAddress` apprende un indirizzo MAC associandolo a una porta specifica. Aggiunge o aggiorna l'indirizzo MAC nella tabella CAM.

```
func (s *Switch) LearnMacAddress(macAddress string,
port int) {
    s.camTable[macAddress] = port
    fmt.Printf("MAC %s learned on port %d\n",
macAddress, port)
}
```

Il metodo `ForwardFrame` simula il processo di inoltro di un frame. Apprende l'indirizzo MAC sorgente e cerca l'indirizzo MAC di destinazione nella tabella CAM. Se l'indirizzo MAC di destinazione è trovato, il frame viene inoltrato alla porta corrispondente. Se non è trovato, il frame viene inondato a tutte le porte tranne quella di ingresso.

```
func (s *Switch) ForwardFrame(srcMac, dstMac string,
port int) {
    s.LearnMacAddress(srcMac, port)
    if outPort, found := s.camTable[dstMac]; found {
        fmt.Printf("Forwarding frame from %s to %s
via port %d\n", srcMac, dstMac, outPort)
    } else {
        fmt.Printf("Flooding frame from %s to %s to
all ports except %d\n", srcMac, dstMac, port)
```

```
    }
}
```

Il metodo `PrintCamTable` stampa la tabella CAM corrente, mostrando quali indirizzi MAC sono stati appresi e a quali porte sono associati.

```
func (s *Switch) PrintCamTable() {
    fmt.Println("\nCurrent CAM Table:")
    for mac, port := range s.camTable {
        fmt.Printf("MAC Address: %s -> Port: %d\n",
mac, port)
    }
}
```

La funzione `GenerateMacAddress` genera un indirizzo MAC casuale utilizzando il package `uuid`.

```
func GenerateMacAddress() string {
    u := uuid.New()
    mac :=
strings.ToUpper(fmt.Sprintf("%02X:%02X:%02X:%02X:%02X
:%02X", u[0], u[1], u[2], u[3], u[4], u[5]))
    return mac
}
```

Ecco l'esempio completo che integra tutti i componenti:

```
package main

import (
    "fmt"
    "strings"
    "github.com/google/uuid"
)

type Switch struct {
    camTable map[string]int // Tabella di
indirizzamento CAM
}

func NewSwitch() *Switch {
    return &Switch{camTable: make(map[string]int)}
}

func (s *Switch) LearnMacAddress(macAddress string,
port int) {
    s.camTable[macAddress] = port
    fmt.Printf("MAC %s learned on port %d\n",
macAddress, port)
}

func (s *Switch) ForwardFrame(srcMac, dstMac string,
port int) {
    s.LearnMacAddress(srcMac, port)
    if outPort, found := s.camTable[dstMac]; found {
        fmt.Printf("Forwarding frame from %s to %s
via port %d\n", srcMac, dstMac, outPort)
    } else {
        fmt.Printf("Flooding frame from %s to %s to
all ports except %d\n", srcMac, dstMac, port)
```

```
    }
}

func (s *Switch) PrintCamTable() {
    fmt.Println("\nCurrent CAM Table:")
    for mac, port := range s.camTable {
        fmt.Printf("MAC Address: %s -> Port: %d\n",
mac, port)
    }
}

func GenerateMacAddress() string {
    u := uuid.New()
    mac :=
strings.ToUpper(fmt.Sprintf("%02X:%02X:%02X:%02X:%02X
:%02X", u[0], u[1], u[2], u[3], u[4], u[5]))
    return mac
}

func main() {
    switchDevice := NewSwitch()

    srcMac1 := GenerateMacAddress()
    dstMac1 := GenerateMacAddress()
    srcMac2 := GenerateMacAddress()
    dstMac2 := GenerateMacAddress()
    srcMac3 := GenerateMacAddress()
    dstMac3 := GenerateMacAddress()

    switchDevice.ForwardFrame(srcMac1, dstMac1, 1)
    switchDevice.ForwardFrame(srcMac2, dstMac1, 2)
    switchDevice.ForwardFrame(srcMac3, dstMac2, 1)
```

```
    switchDevice.PrintCamTable()
}
```

Conclusione

Questo esempio mostra come uno switch apprende gli indirizzi MAC e utilizza la tabella CAM per inoltrare i frame ai porti corretti, migliorando l'efficienza della rete. La simulazione in Go fornisce una rappresentazione pratica di come funzionano gli switch di rete, offrendo un'ottima base per comprendere i concetti fondamentali dell'indirizzamento e della commutazione degli indirizzi MAC.