

gtcp/TCPgtcp/gtcp

gtcp

```
(16bit)|()
```

1. $16(2)^2$
2. $0xFFFF = 65535 \text{ bytes} = 64 \text{ KB}$

gtcpgtcp/



Content Menu

-
-
-
- [1](#)
- [2](#)

<https://pkg.go.dev/github.com/gogf/gf/v2/net/gtcp>

```
type Conn
func (c *Conn) SendPkg(data []byte, option ...PkgOption) error
func (c *Conn) SendPkgWithTimeout(data []byte, timeout time.Duration,
option ...PkgOption) error
func (c *Conn) SendRecvPkg(data []byte, option ...PkgOption) ([]byte,
error)
func (c *Conn) SendRecvPkgWithTimeout(data []byte, timeout time.
Duration, option ...PkgOption) ([]byte, error)
func (c *Conn) RecvPkg(option ...PkgOption) (result []byte, err error)
func (c *Conn) RecvPkgWithTimeout(timeout time.Duration, option ...
PkgOption) ([]byte, error)
```

Pkg

PkgOption

```
//
type PkgOption struct {
    HeaderSize int    // (24)
    MaxDataSize int    // (byte)2(65535 byte)
    Retry      Retry //
}
```

```

package main

import (
    "fmt"
    "github.com/gogf/gf/v2/net/gtcp"
    "github.com/gogf/gf/v2/os/glog"
    "github.com/gogf/gf/v2/util/gconv"
    "time"
)

func main() {
    // Server
    go gtcp.NewServer("127.0.0.1:8999", func(conn *gtcp.Conn) {
        defer conn.Close()
        for {
            data, err := conn.RecvPkg()
            if err != nil {
                fmt.Println(err)
                break
            }
            fmt.Println("receive:", data)
        }
    }).Run()

    time.Sleep(time.Second)

    // Client
    conn, err := gtcp.NewConn("127.0.0.1:8999")
    if err != nil {
        panic(err)
    }
    defer conn.Close()
    for i := 0; i < 10000; i++ {
        if err := conn.SendPkg([]byte(gconv.String(i))); err !=
nil {
            glog.Error(err)
        }
        time.Sleep(1*time.Second)
    }
}

```

```

receive: [48]
receive: [49]
receive: [50]
receive: [51]
...

```

2

CPUJS0N/

 protobuf/

1. types/types.go

```

package types

import "github.com/gogf/gf/v2/frame/g"

type NodeInfo struct {
    Cpu      float32 // CPU(%)
    Host     string  //
    Ip       g.Map   // IP()
    MemUsed  int     // (byte)
    MemTotal int     // (byte)
    Time     int     // ()
}

```

2. gtcp_monitor_server.go

```

package main

import (
    "encoding/json"
    "github.com/gogf/gf/v2/net/gtcp"
    "github.com/gogf/gf/v2/os/glog"
    "github.com/gogf/gf/.example/net/gtcp/pkg_operations/monitor"
    /types"
)

func main() {
    //
    gtcp.NewServer("127.0.0.1:8999", func(conn *gtcp.Conn) {
        defer conn.Close()
        for {
            data, err := conn.RecvPkg()
            if err != nil {
                if err.Error() == "EOF" {
                    glog.Println("client closed")
                }
                break
            }
            info := &types.NodeInfo{}
            if err := json.Unmarshal(data, info); err != nil {
                glog.Errorf("invalid package structure: %s", err.
Error())
            } else {
                glog.Println(info)
                conn.SendPkg([]byte("ok"))
            }
        }
    }).Run()
}

```

3. gtcp_monitor_client.go

```

package main

import (
    "encoding/json"
    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/net/gtcp"
    "github.com/gogf/gf/v2/os/glog"
    "github.com/gogf/gf/v2/os/gtime"
    "github.com/gogf/gf/.example/net/gtcp/pkg_operations/monitor"
    /types"
)

func main() {
    //
    conn, err := gtcp.NewConn("127.0.0.1:8999")
    if err != nil {
        panic(err)
    }
    defer conn.Close()
    // JSON
    info, err := json.Marshal(types.NodeInfo{
        Cpu      : float32(66.66),
        Host     : "localhost",
        Ip       : g.Map {
            "eth0" : "192.168.1.100",
            "eth1" : "114.114.10.11",
        },
        MemUsed  : 15560320,
        MemTotal : 16333788,
        Time     : int(gtime.Timestamp()),
    })
    if err != nil {
        panic(err)
    }
    // SendRecvPkg
    if result, err := conn.SendRecvPkg(info); err != nil {
        if err.Error() == "EOF" {
            glog.Println("server closed")
        }
    } else {
        glog.Println(string(result))
    }
}

```

4.

-

2019-05-03 13:33:25.710 ok

-

2019-05-03 13:33:25.710 &{66.66 localhost map[eth1:
114.114.10.11 eth0:192.168.1.100] 15560320 16333788 1556861605}
2019-05-03 13:33:25.710 client closed