`accept()`able Strategies for Improving Web Server Performance

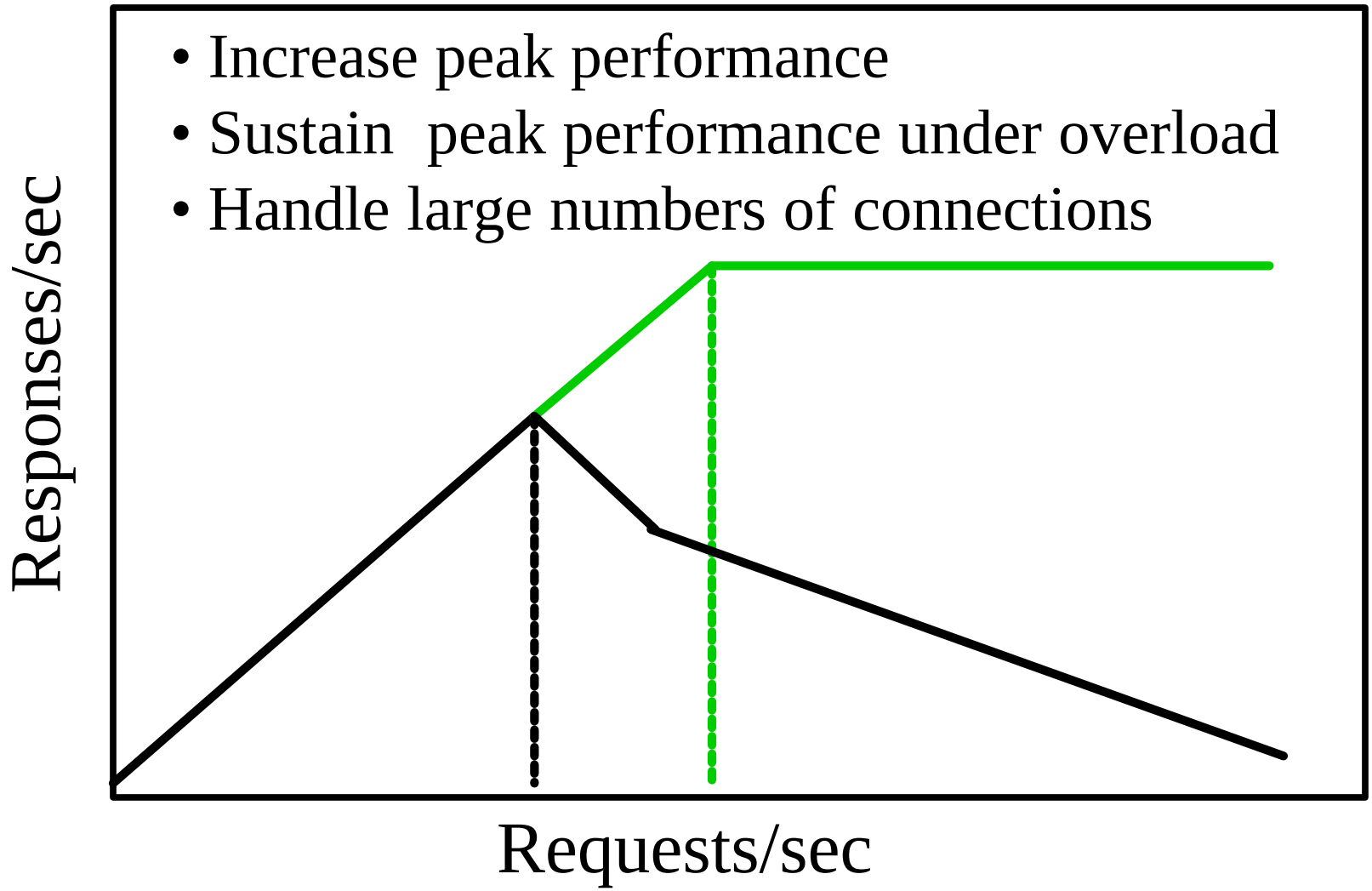
Tim Brecht, David Pariag,

and Louay Gammo

University of Waterloo

`http://cs.uwaterloo.ca/~brecht`

The Problem and Goals



Better understand apps and interaction with kernel

Results

- Examine how different servers **`accept()`** connections
- Increased throughput:
  - **`userver`** **(39-71%)** user-mode, event-driven
  - **`Knot`** **(0-32%)** user-mode, thread-based
  - **`TUX`** **(19-36%)** kernel-mode, event-driven

- **`userver`** performance comparable to **`TUX`**

Some Related Work

- **multi-accept** [Chandra & Mosberger 2001]
  - workload: requesting single 1 byte file
  - call **`accept`** repeatedly until it fails

- **kernel-mode vs user-mode** [Joubert et al. 2001]
  - TUX nearly 2 x best user-mode server

Phases in the Event-Driven userver

Get Events Phase (figure out what to do)

```
get events
  select()
```

Accept Phase

```
get new connections
  n = accept_loop(limit)
```

Work Phase

```
process requests / connections
  read()    write()/sendfile()
```

Phases in the Event-Driven userver

Get Events Phase (figure out what to do)

```
get events
  select()
```

Accept Phase

```
get new connections
  n = accept_loop(limit)
```

Work Phase

```
process requests / connections
  read()    write()/sendfile()
```

Lower Rate 1 10 Inf Higher Rate

Knot (each thread)

```
/* Events are hidden in thread library */
while (1) {
   /* Accept Phase */
   accept_loop(limit);

   /* Work Phase */
   foreach (connection) {
      process_client();
      close();
   }
}
```

Knot (each thread)

```
/* Events are hidden in thread library */
while (1) {
   /* Accept Phase */
   accept_loop(limit);

   /* Work Phase */
   foreach (connection) {
      process_client();
      close();
   }
}
```

Lower Rate 1 10 Inf Higher Rate

TUX

```
/* No get event phase required  (kernel-mode) */
while (1) {
   /* Accept Phase */
   if (accepts_pending()) {
     accept_requests(limit)
   }
   /* Work Phase */
   if (requests_pending()) {
     process_requests()
   }
}
```

TUX

```
/* No get event phase required  (kernel-mode) */
while (1) {
   /* Accept Phase */
   if (accepts_pending()) {
     accept_requests(limit)
   }
   /* Work Phase */
   if (requests_pending()) {
     process_requests()
   }
}
```

Lower Rate **Inf** **10** **1** Higher Rate

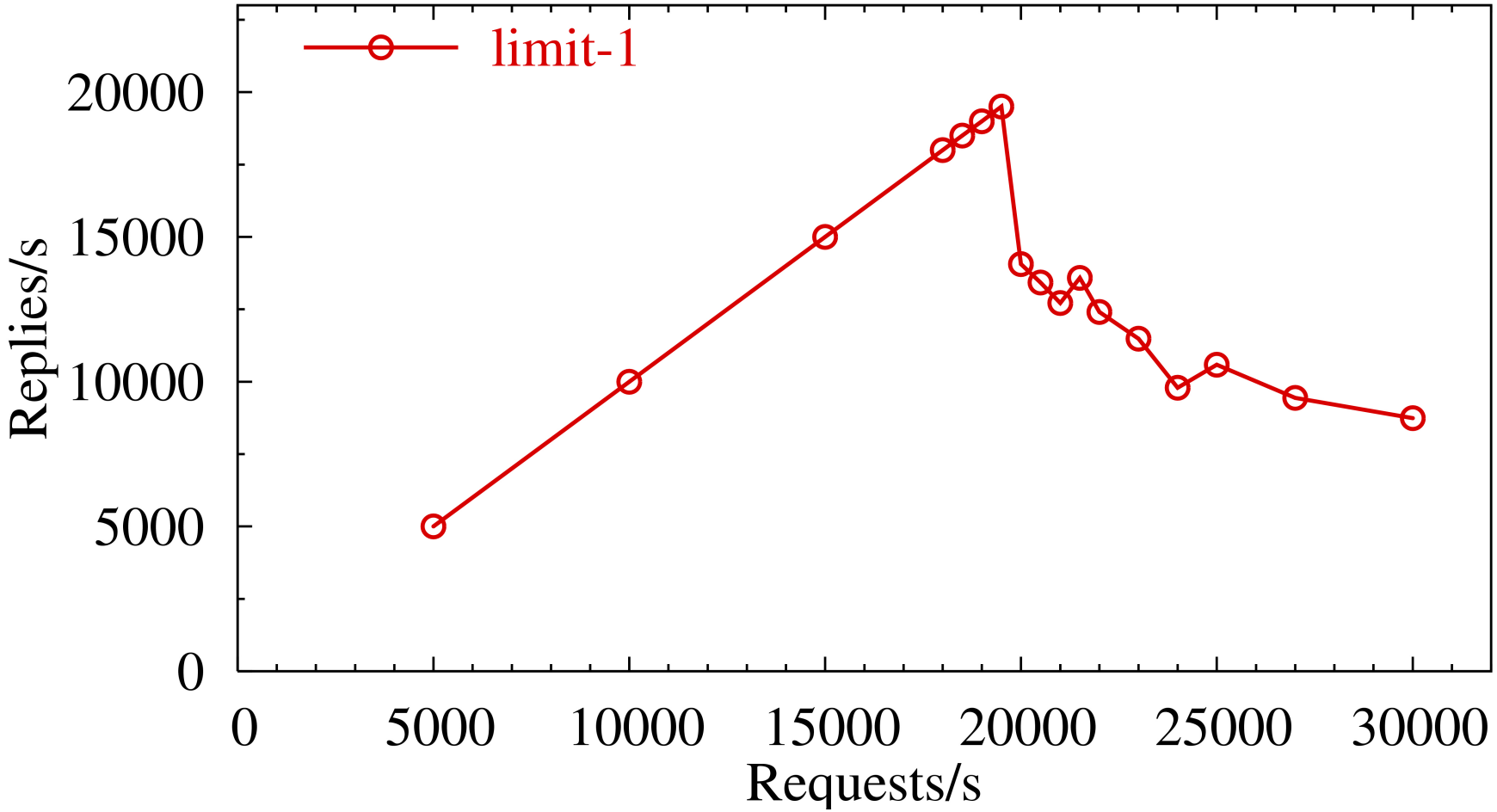
Environment

- **Servers**
  - **`userver, Knot, TUX`**
- **Hardware**
  - 2.4 GHz Xeon, 1 GB RAM, 2 x 1 Gbps Enet
  - 4 clients connected to each 1 Gbps subnet
- **Workload**
  - **One packet workload**

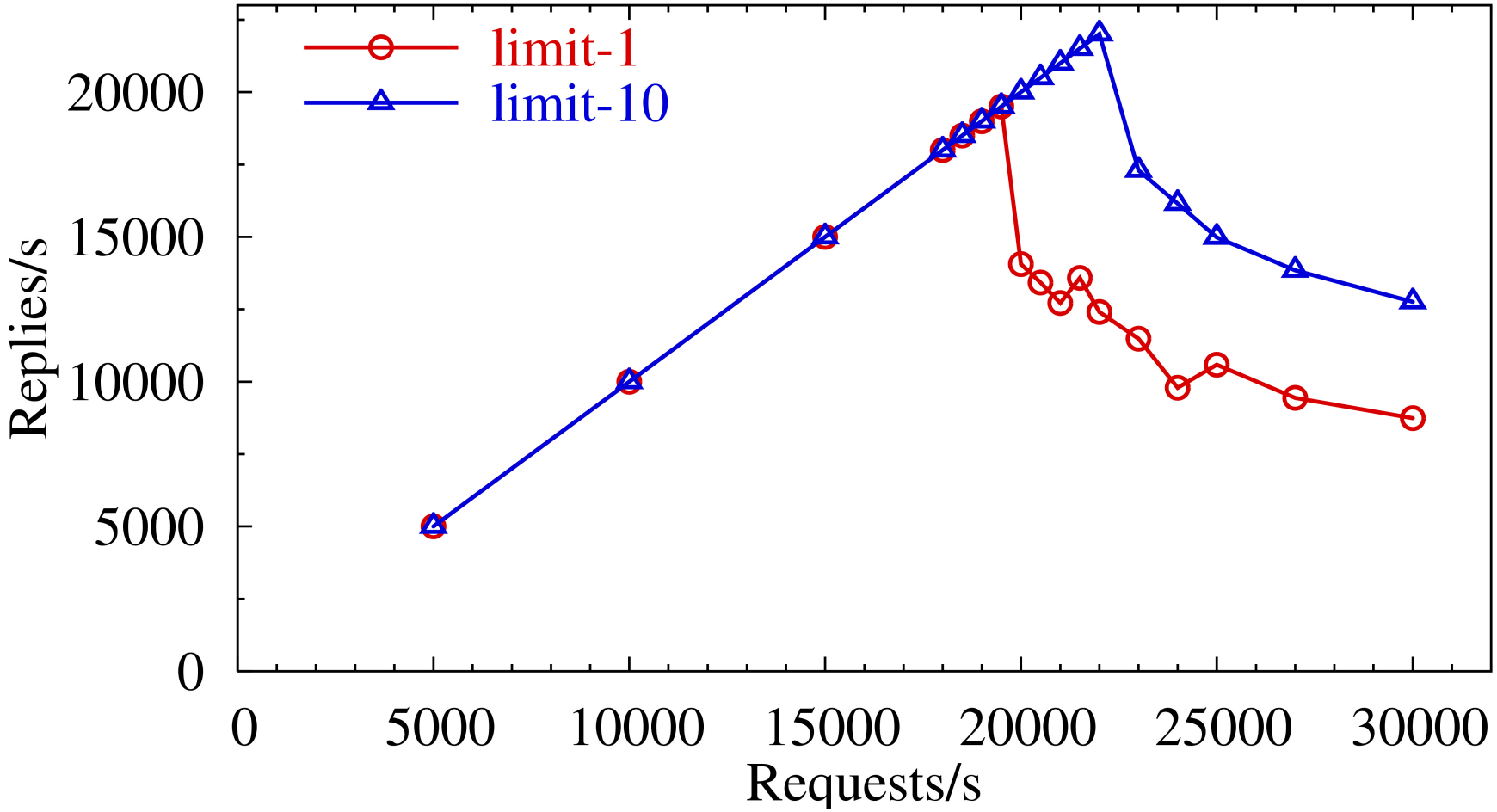
    hammers on OS. CNN.com and upstream on 9/11
  - **Static SPECWeb99-like**

    fits in memory (avoid differences in caching)
  - **`httperf`** can generate overload conditions

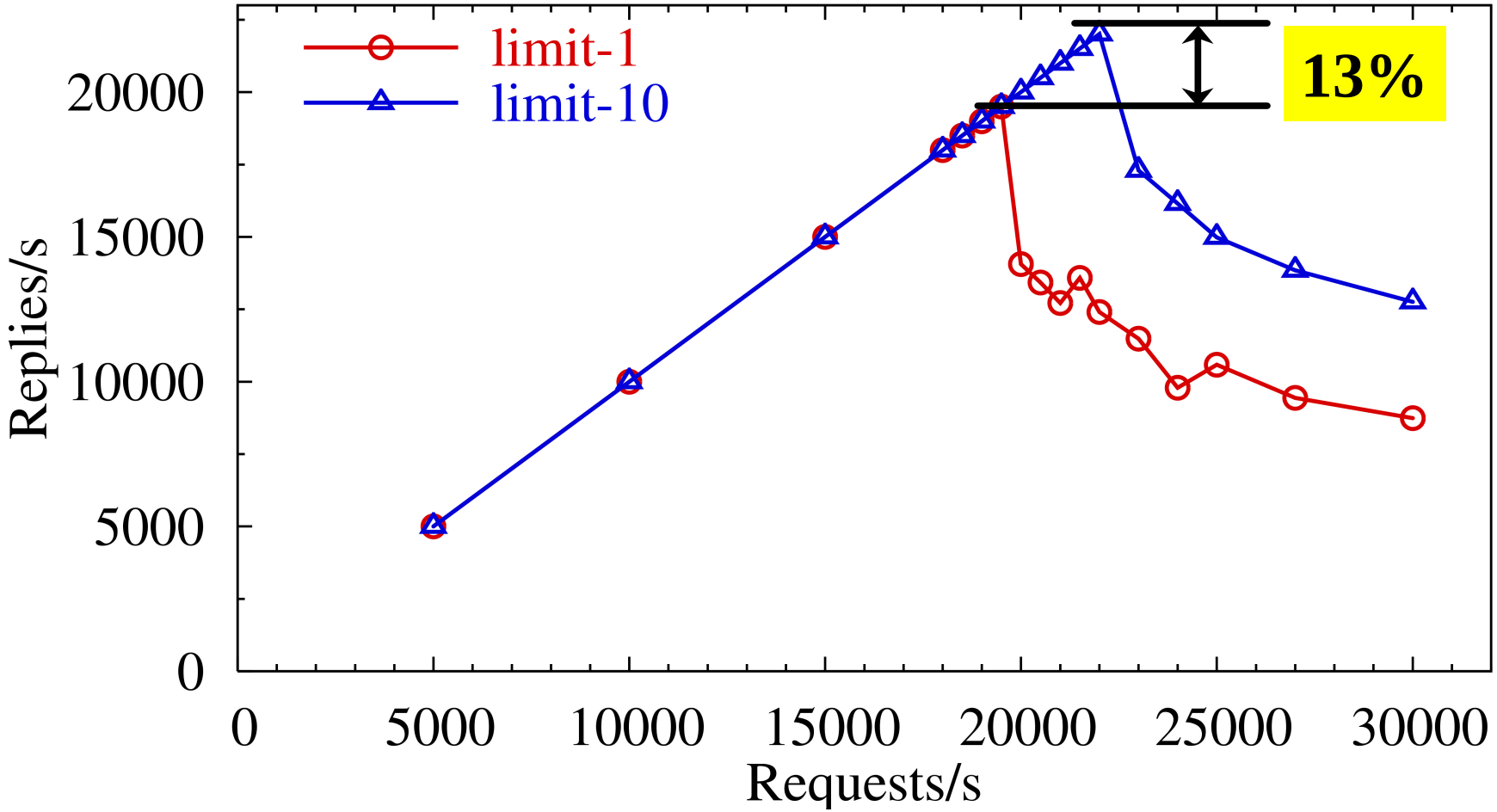
Performance of the userver (one packet)



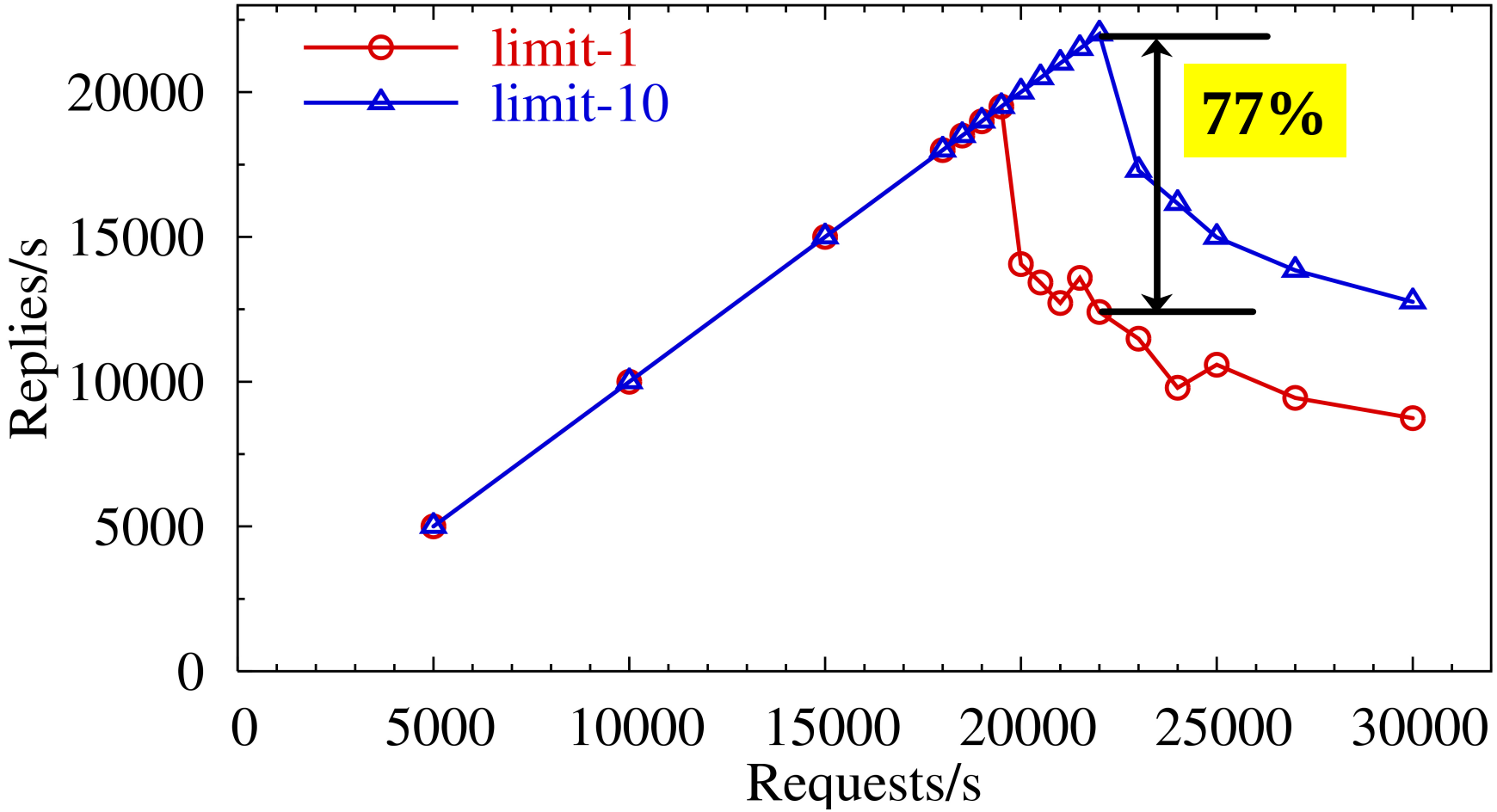
Performance of the userver (one packet)



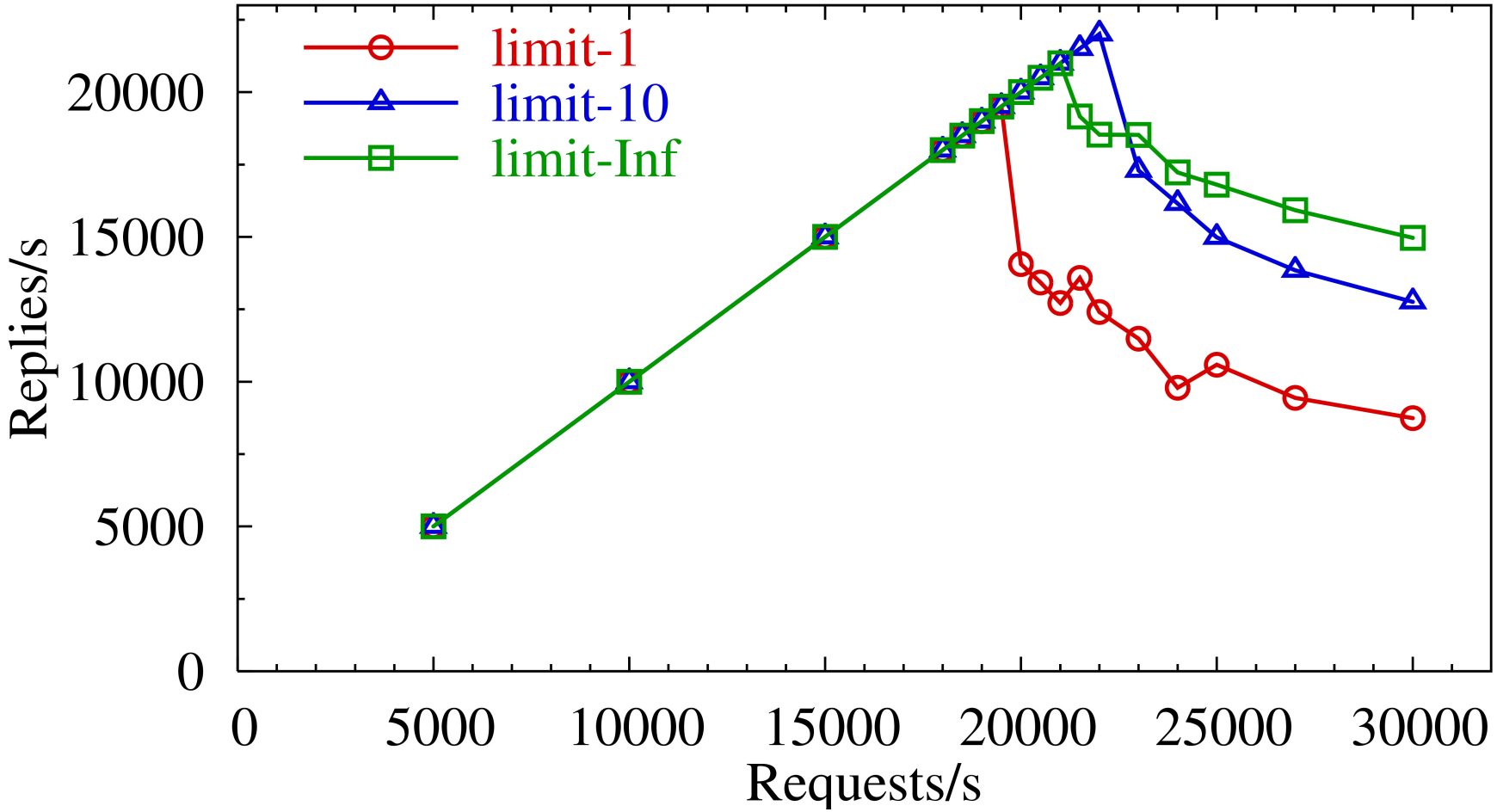
Performance of the userver (one packet)



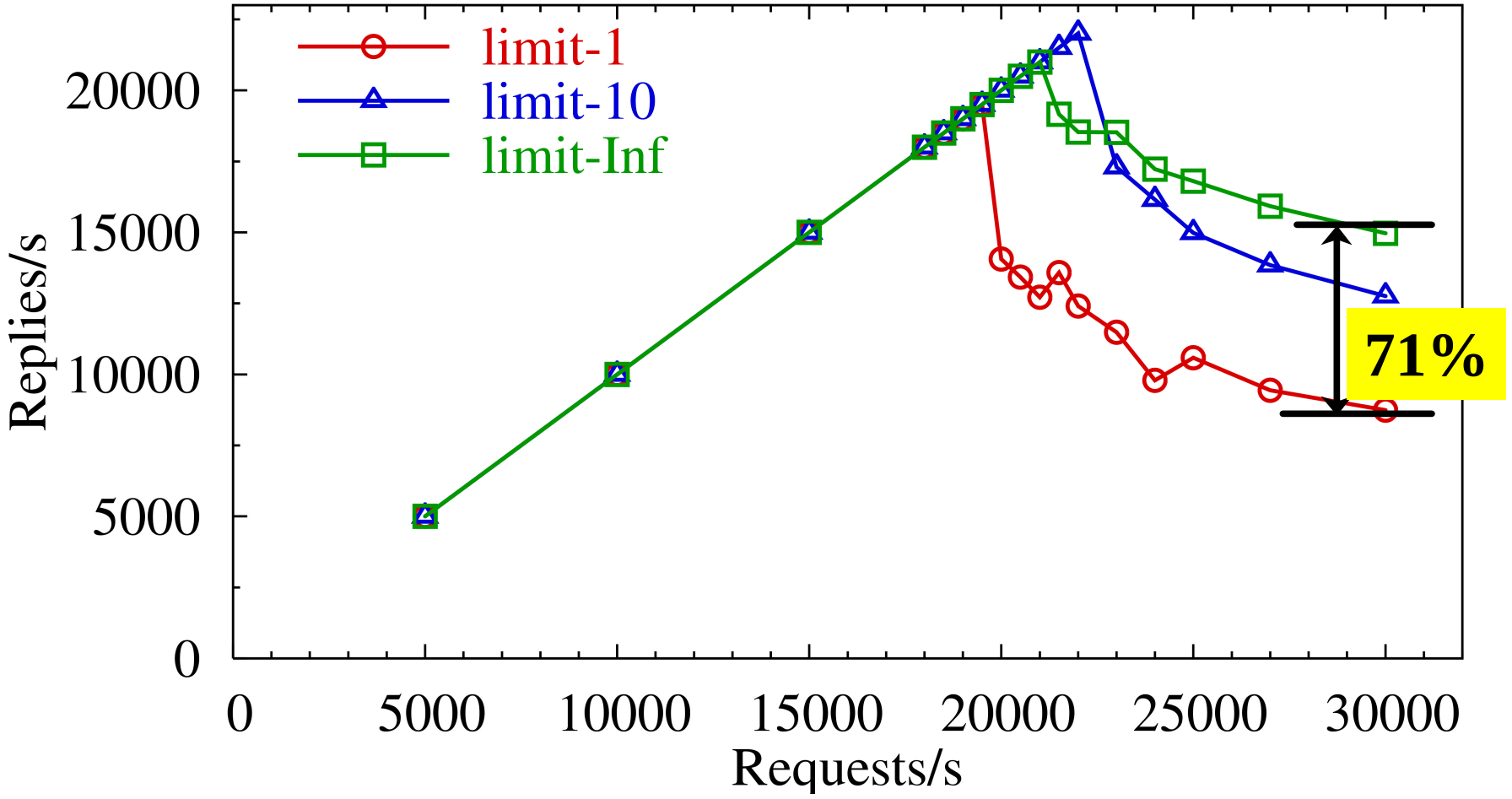
Performance of the userver (one packet)



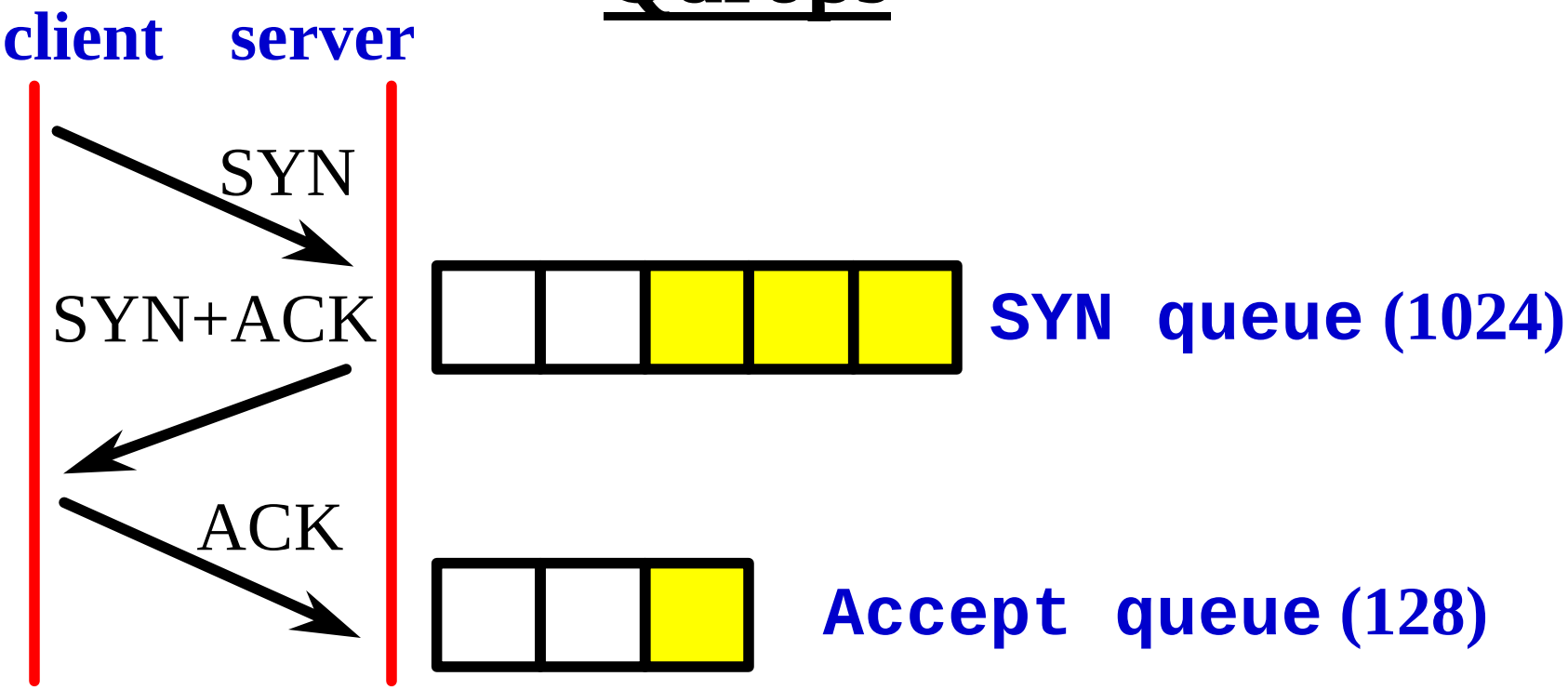
Performance of the userver (one packet)



Performance of the userver (one packet)

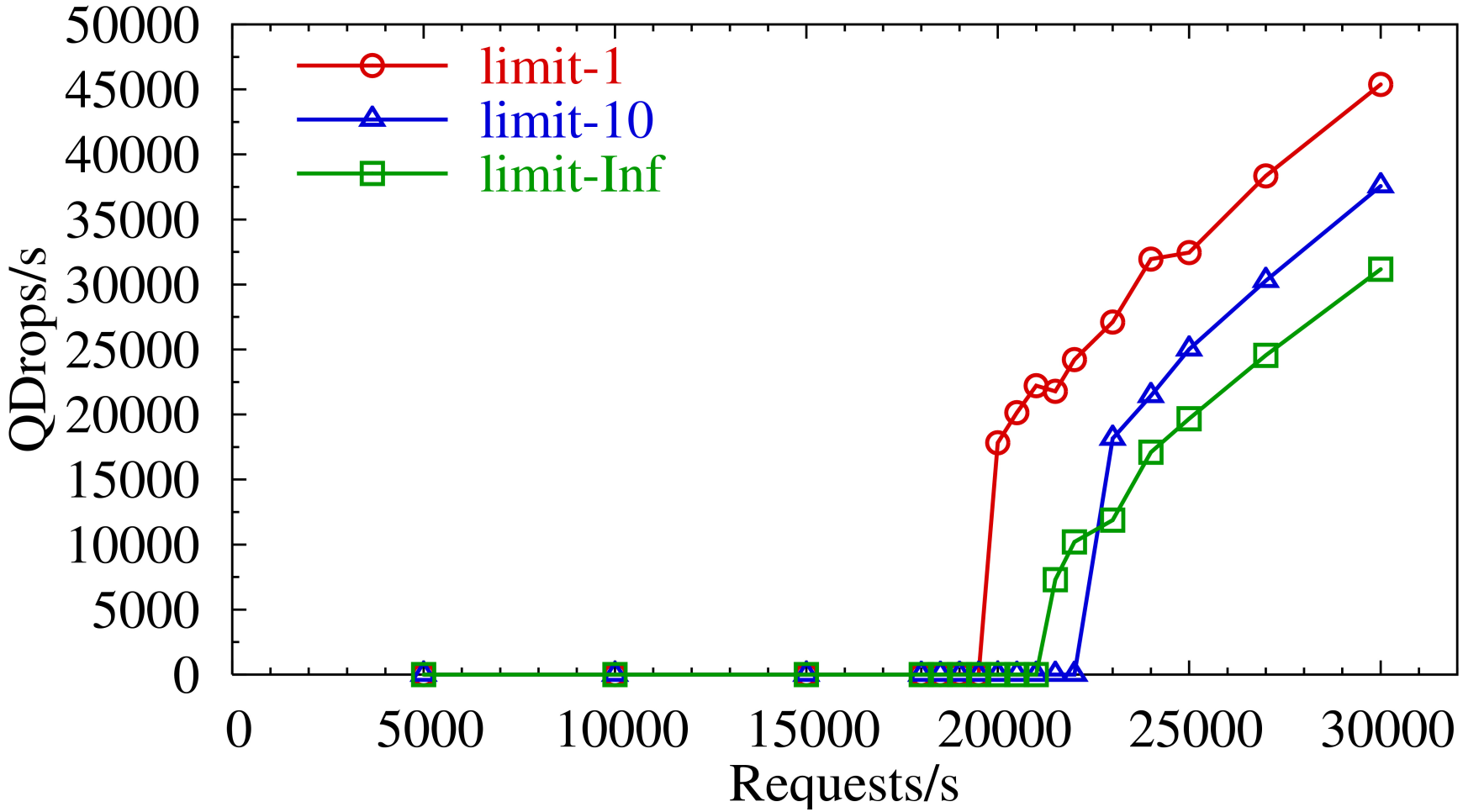


Qdrops

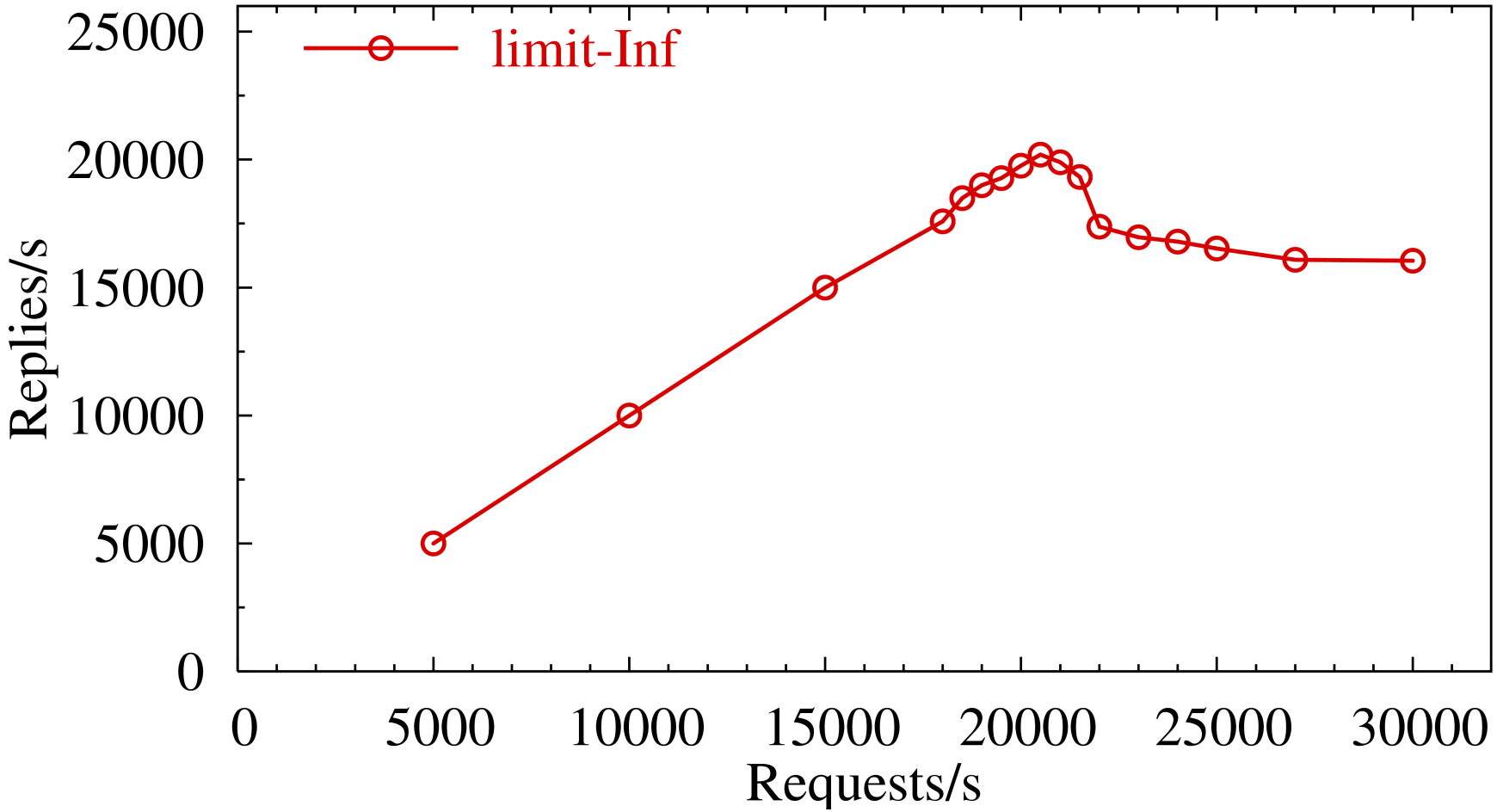


- When SYN arrives **SYN queue** is > 3/4 full
- When SYN arrives **Accept queue** is full
- When ACK arrives **Accept queue** is full

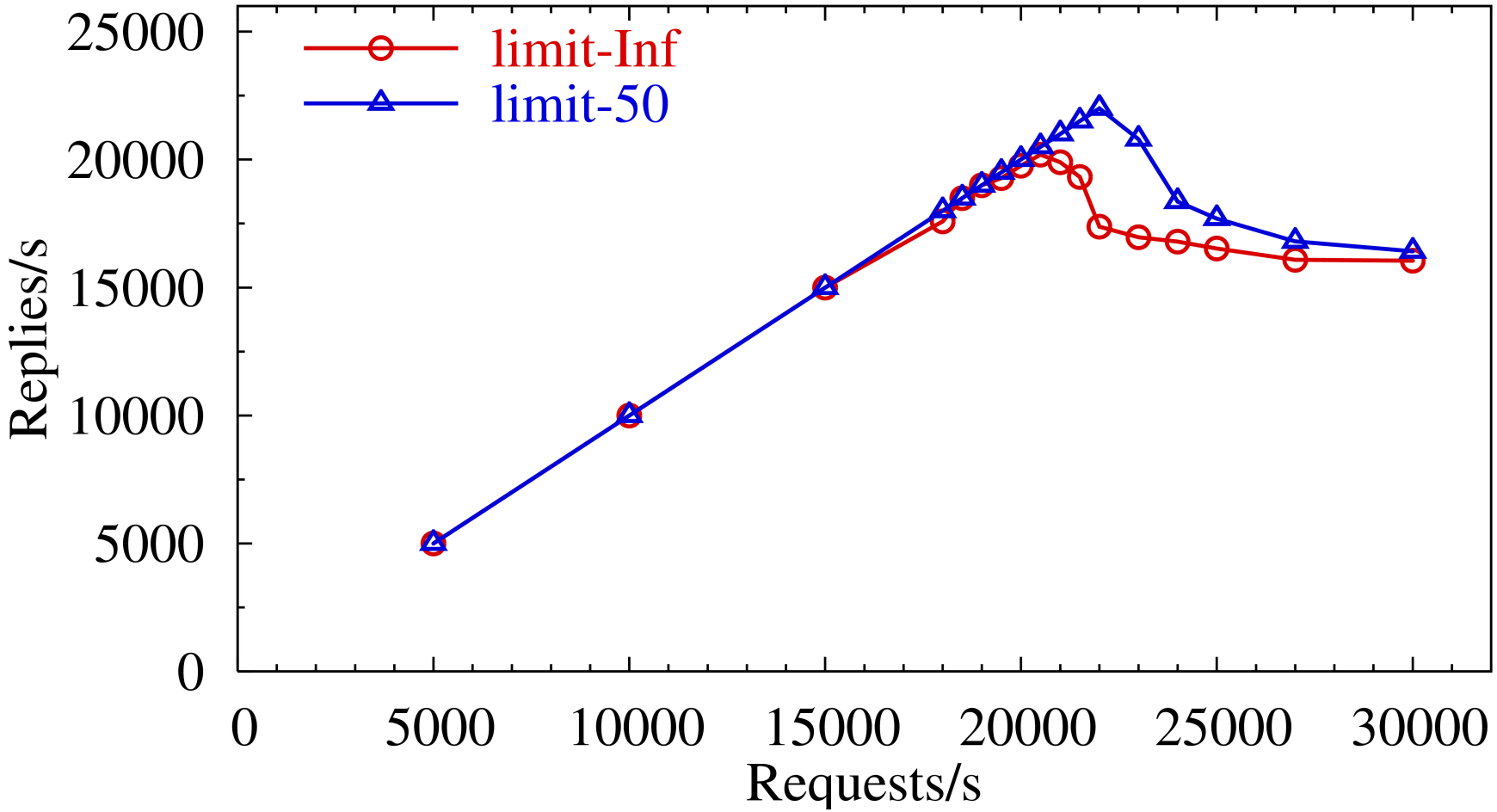
Qdrops for userver (one packet)



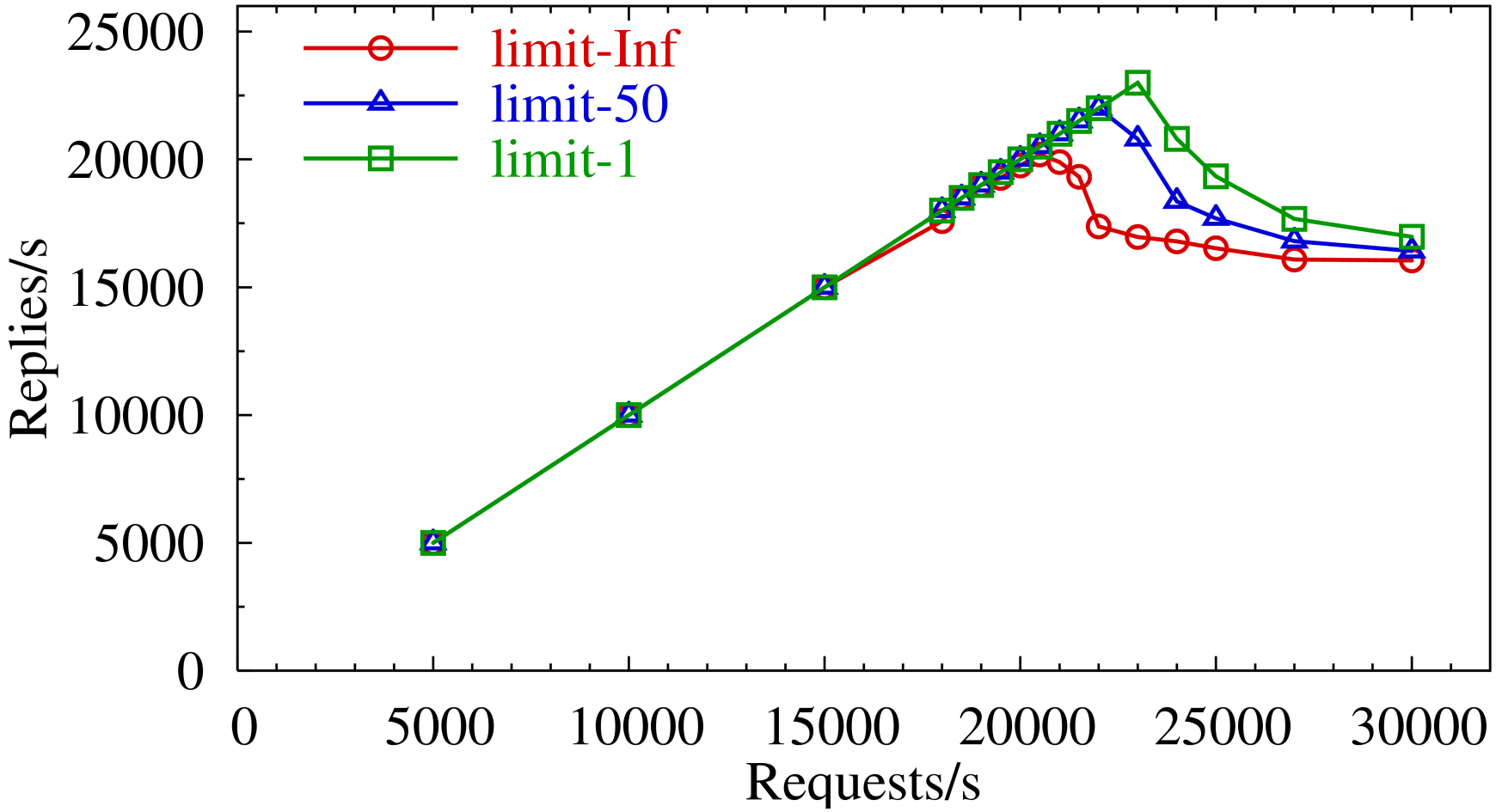
Performance of TUX (one packet)



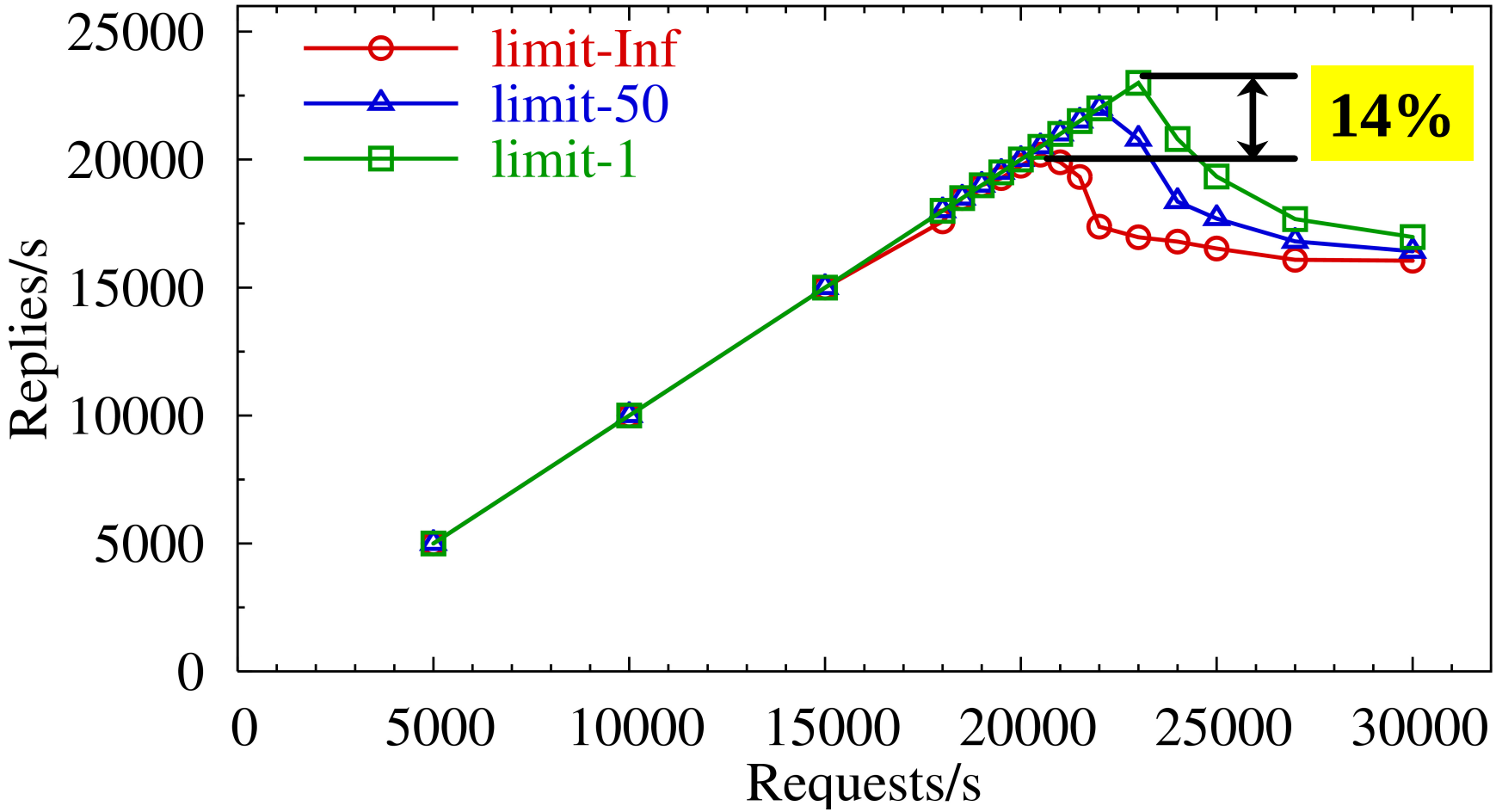
Performance of TUX (one packet)



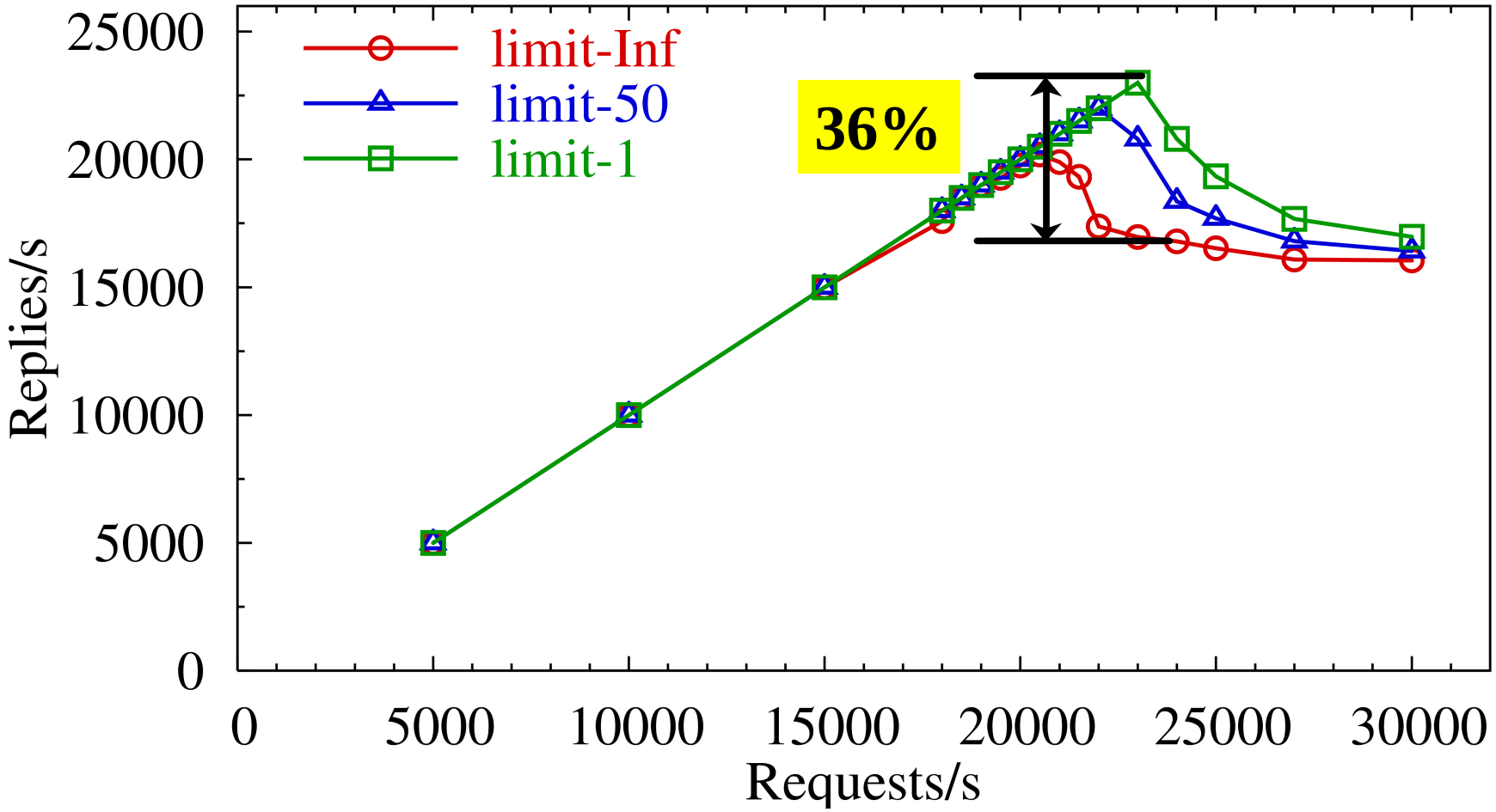
Performance of TUX (one packet)



Performance of TUX (one packet)

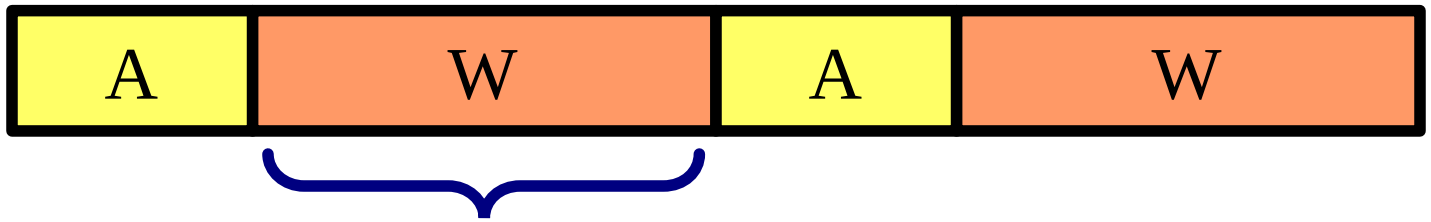


Performance of TUX (one packet)



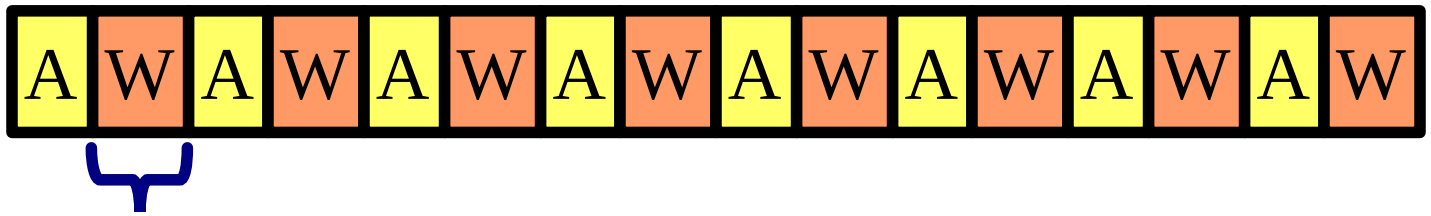
Accept-Limit Impact on TUX

A = 407 W = 10,740 @ 26,000 req/sec



limit-Inf

A = 1 W = 48



limit-1

- **lower limit => more frequent accepts without overhead**
- **lower limit => less opportunity for queues to overflow**

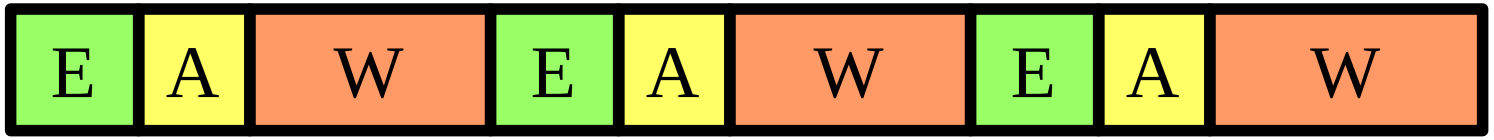
Accept-Limit Impact on the userver

A=1 W=22 @ 26,000 req/sec

Accept-Limit Impact on the userver

A=1 W=22 @ 26,000 req/sec

A=112 W=886

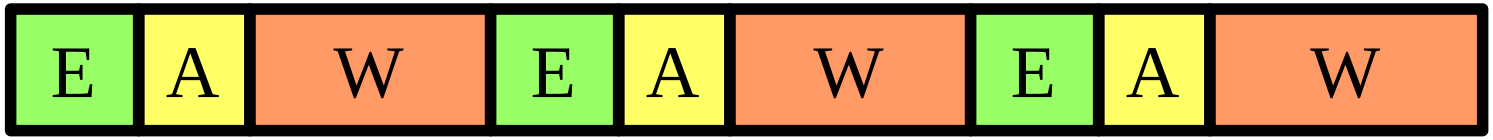


Accept-Limit Impact on the userver

A=1 W=22 @ 26,000 req/sec

limit-1

A=112 W=886



limit-Inf

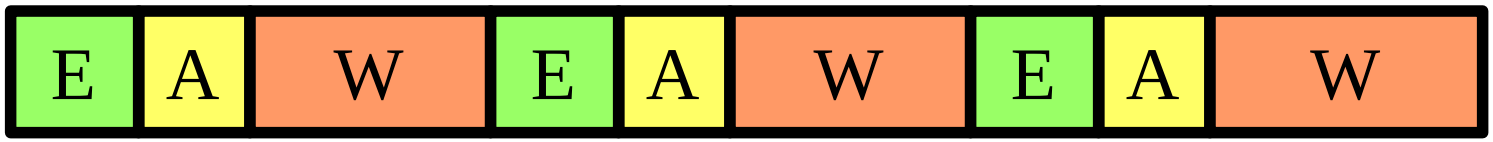
#E's = 184322

limit-1

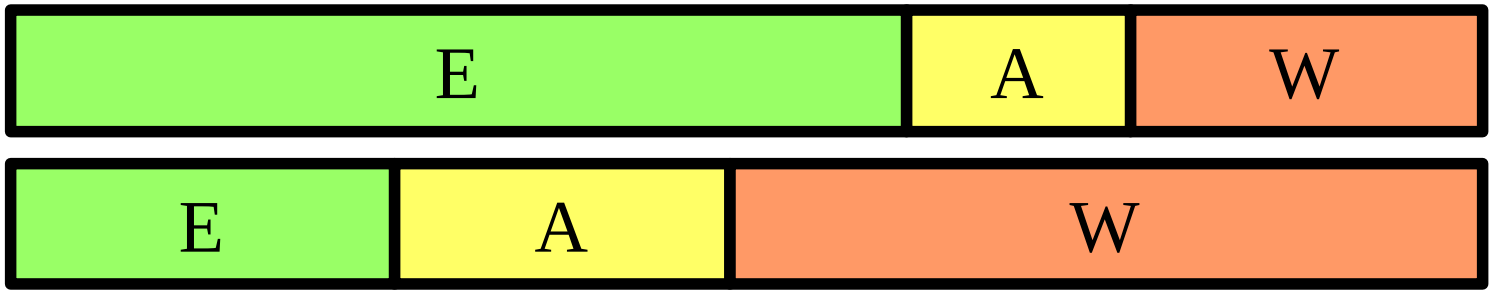
Accept-Limit Impact on the userver

A=1 W=22 @ 26,000 req/sec

A=112 W=886

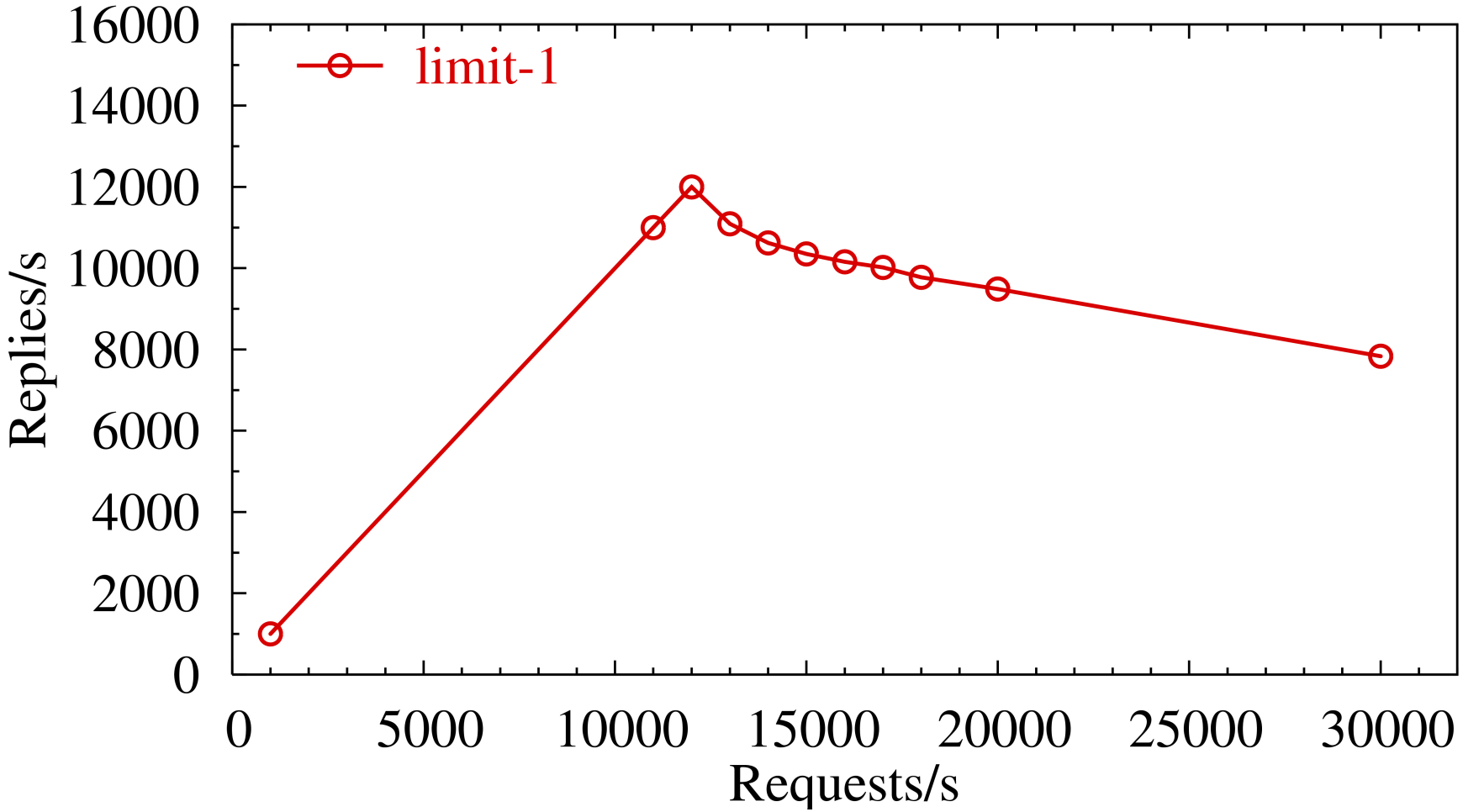


#E's = 184322

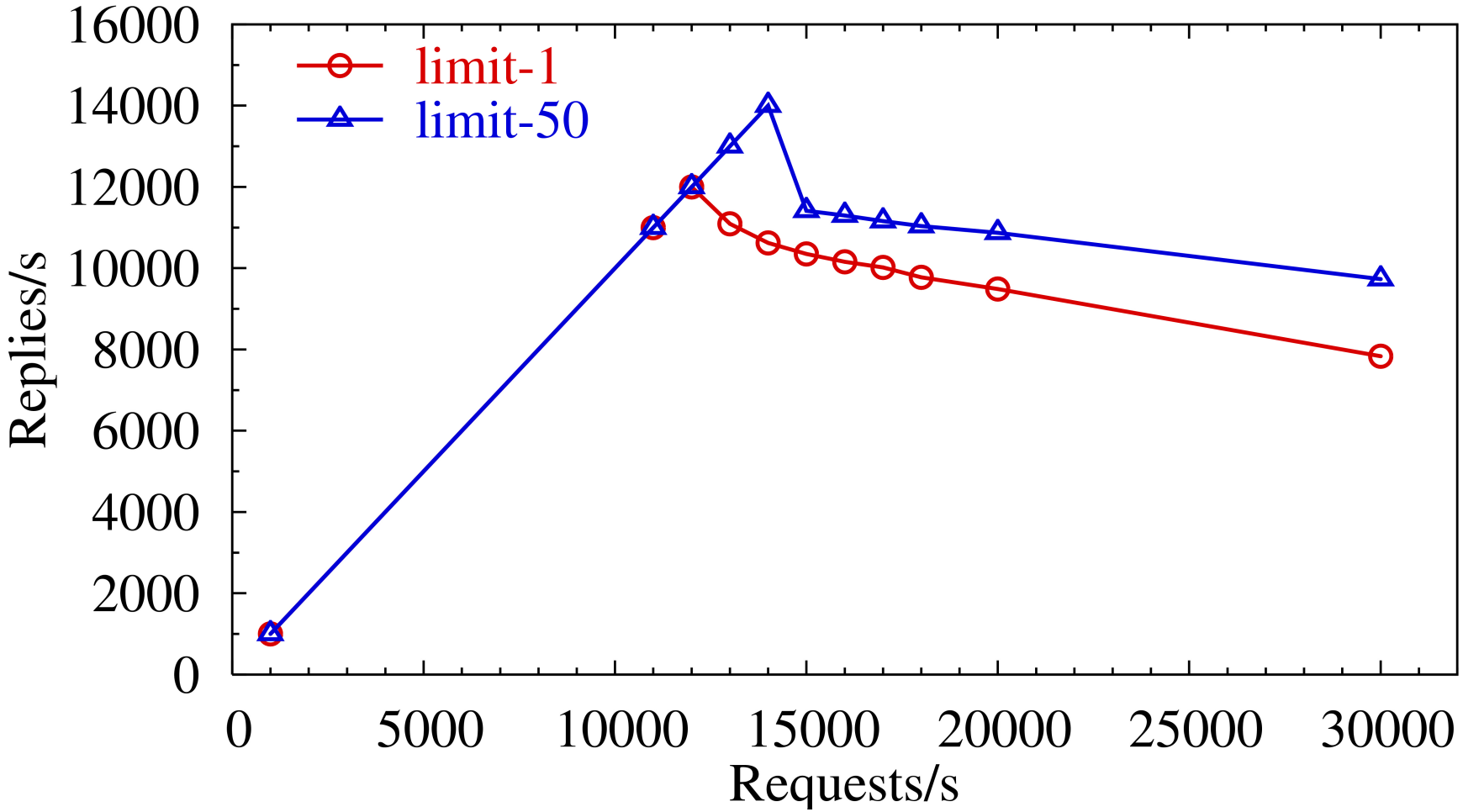


#E's = 5201

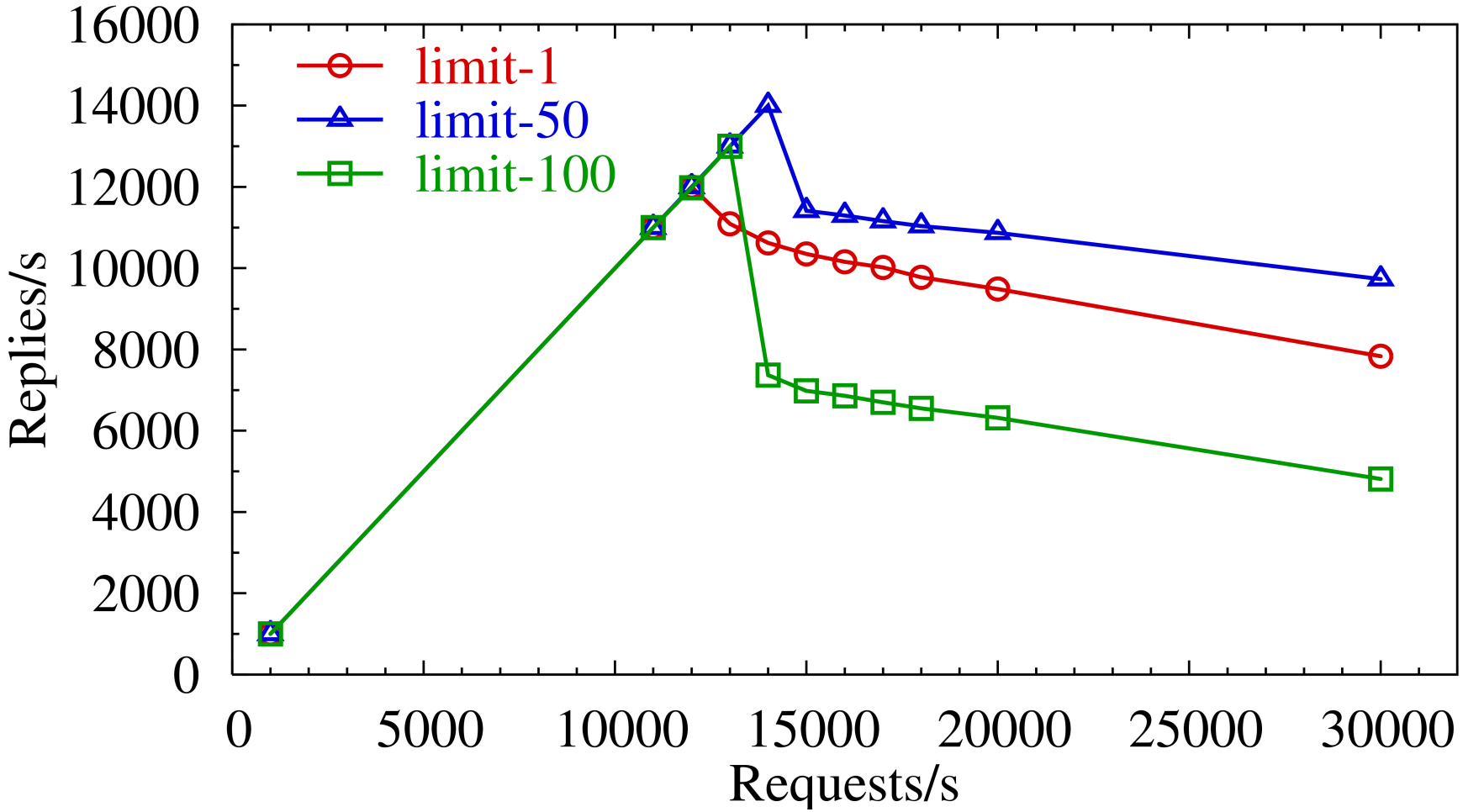
Performance of Knot (one packet)



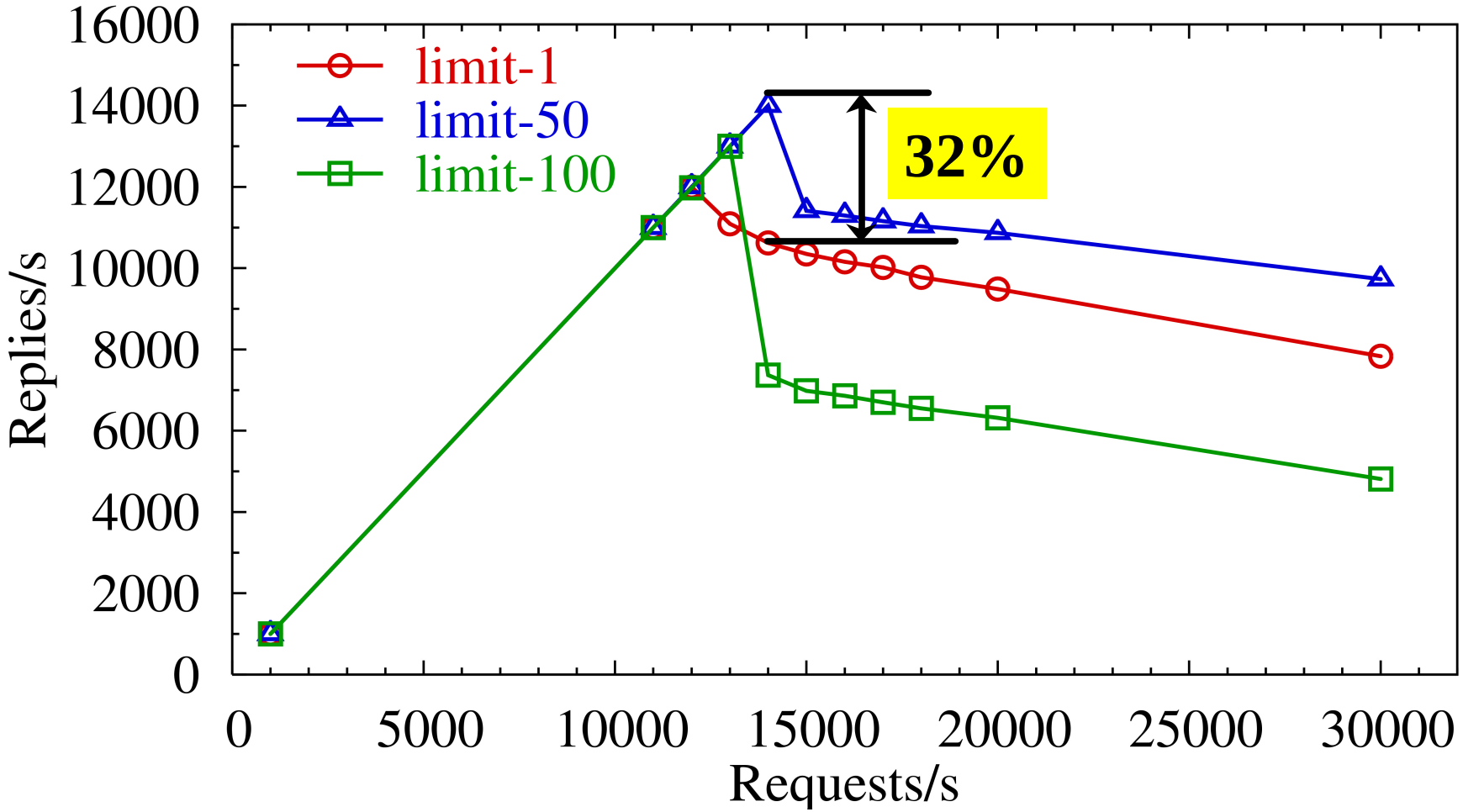
Performance of Knot (one packet)



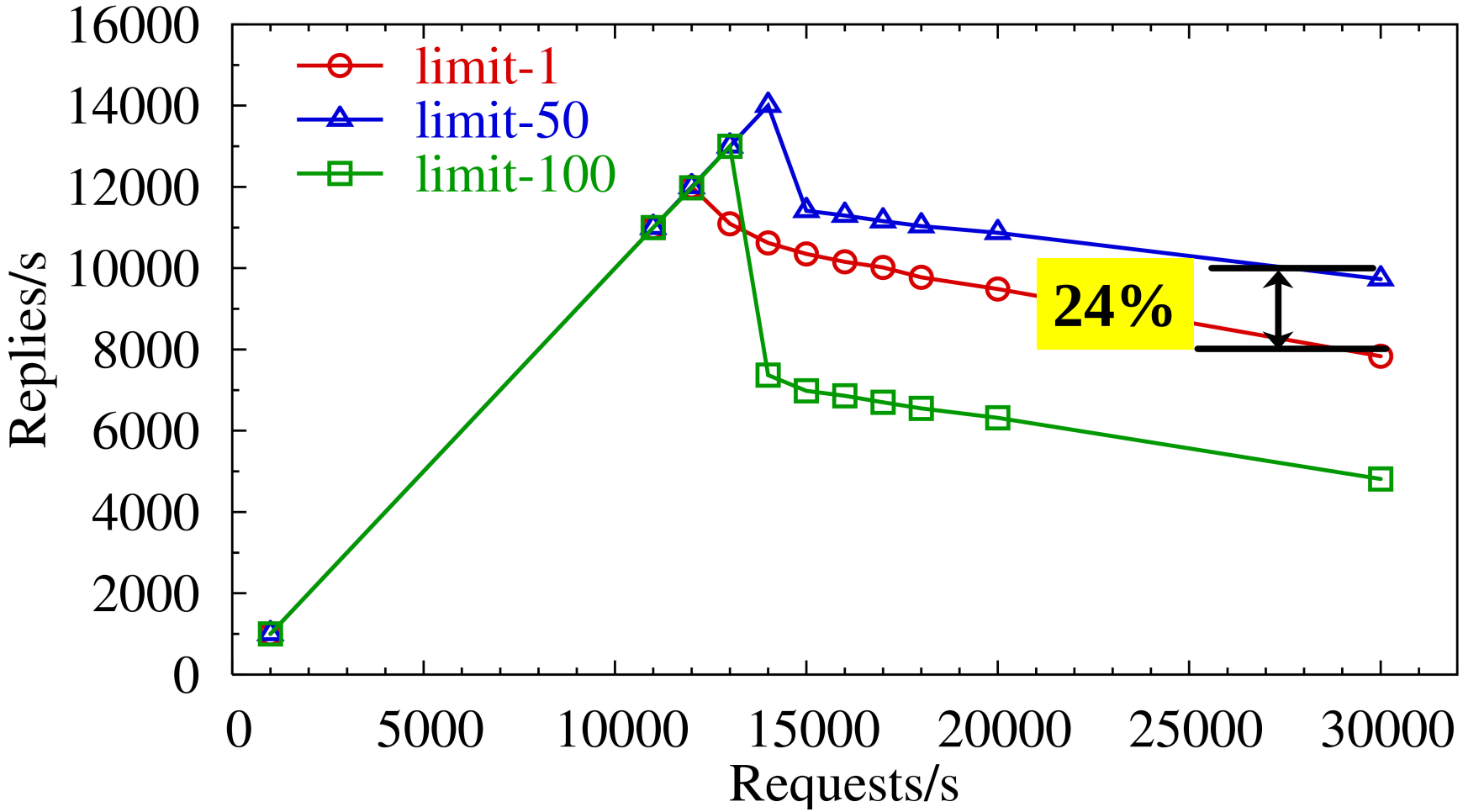
Performance of Knot (one packet)



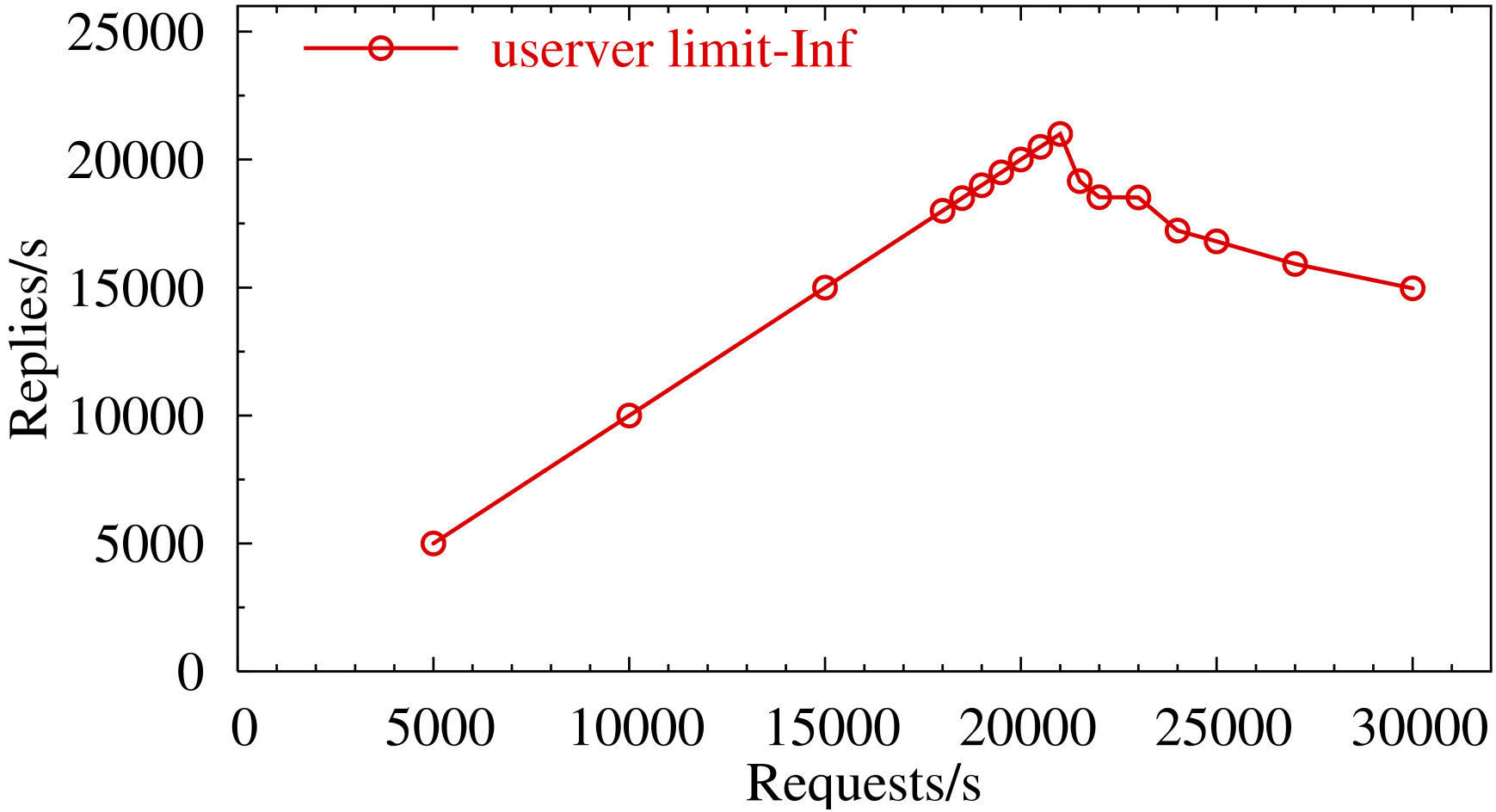
Performance of Knot (one packet)



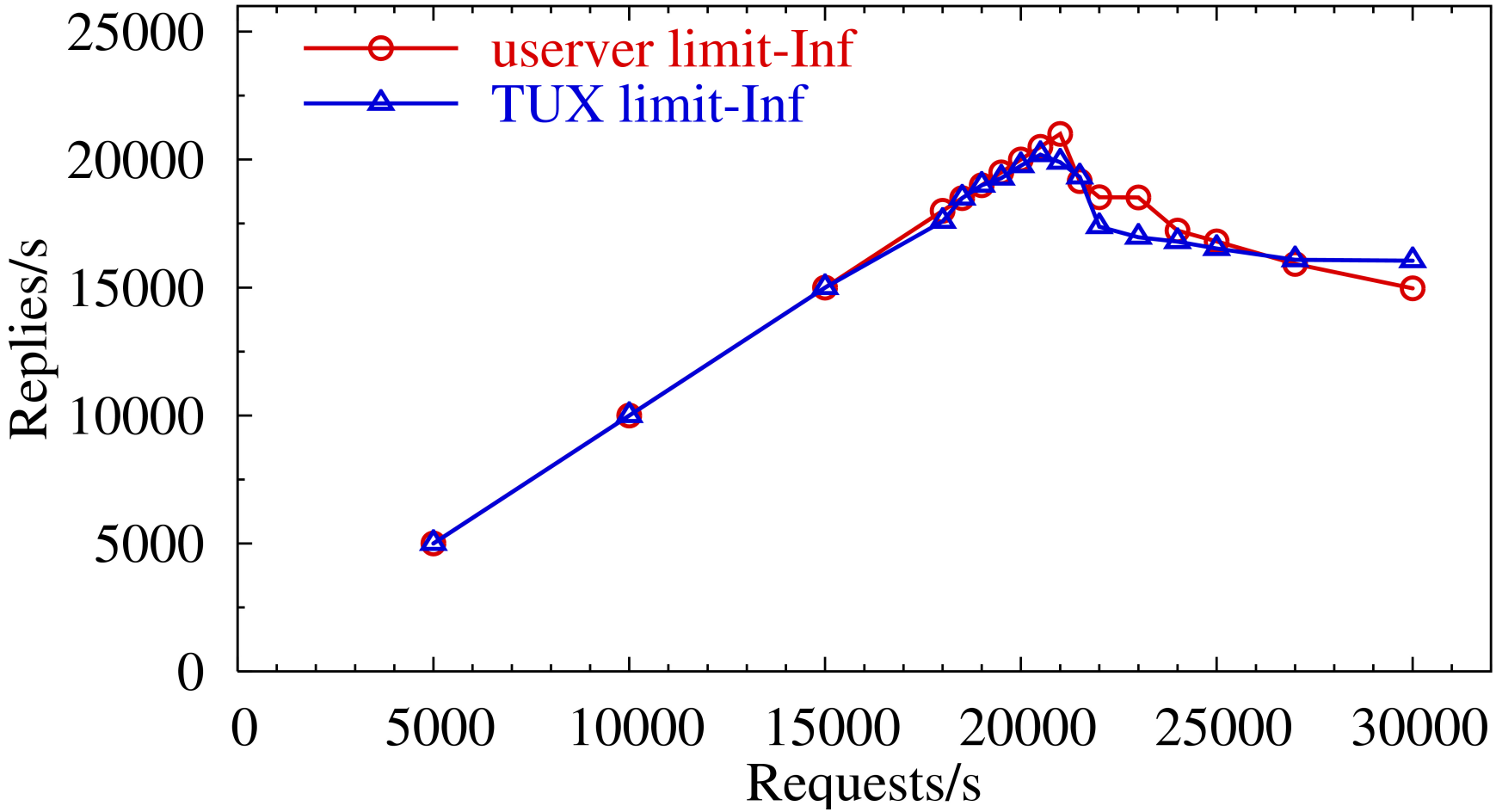
Performance of Knot (one packet)



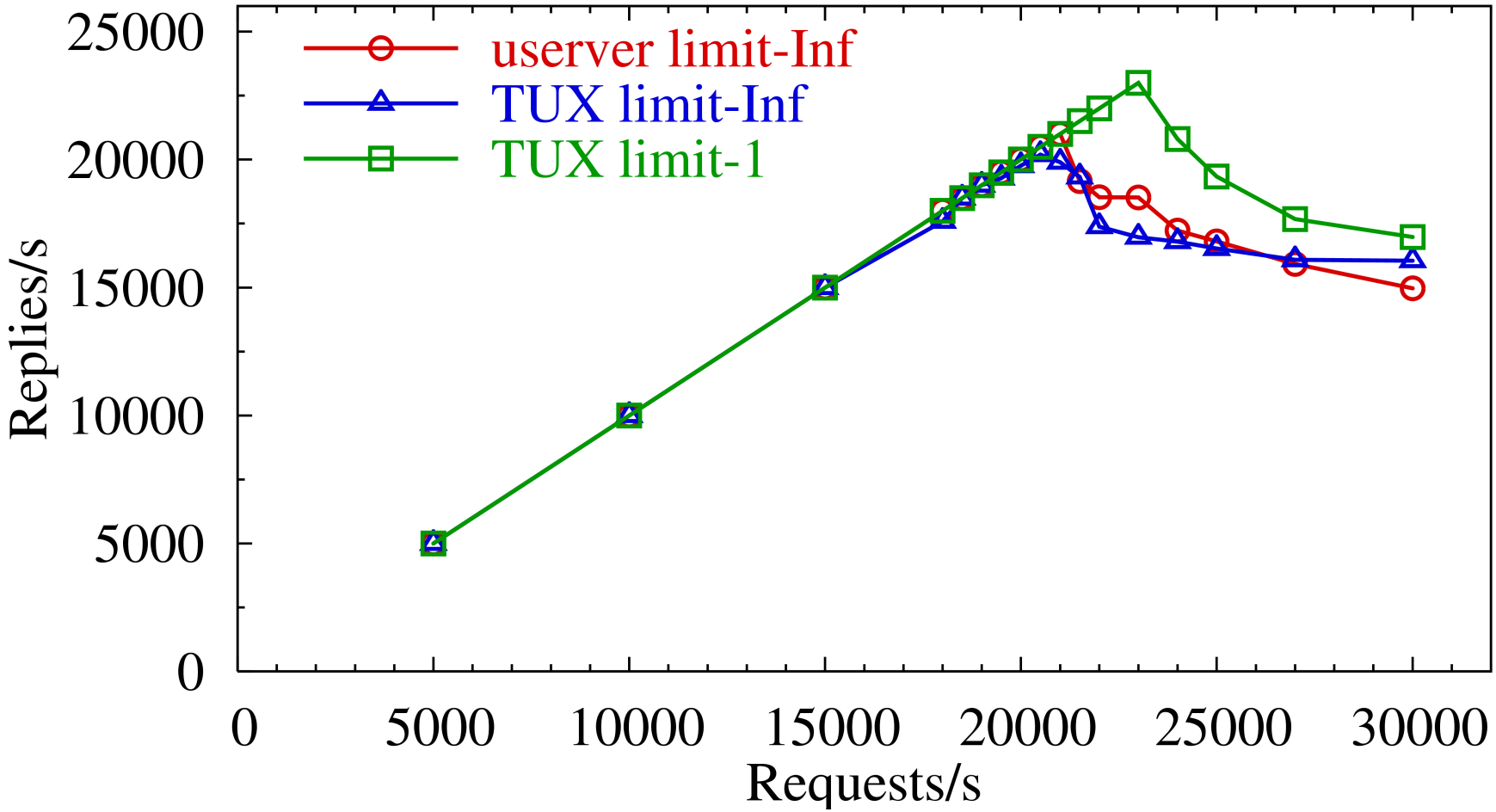
TUX vs the userver (one packet)



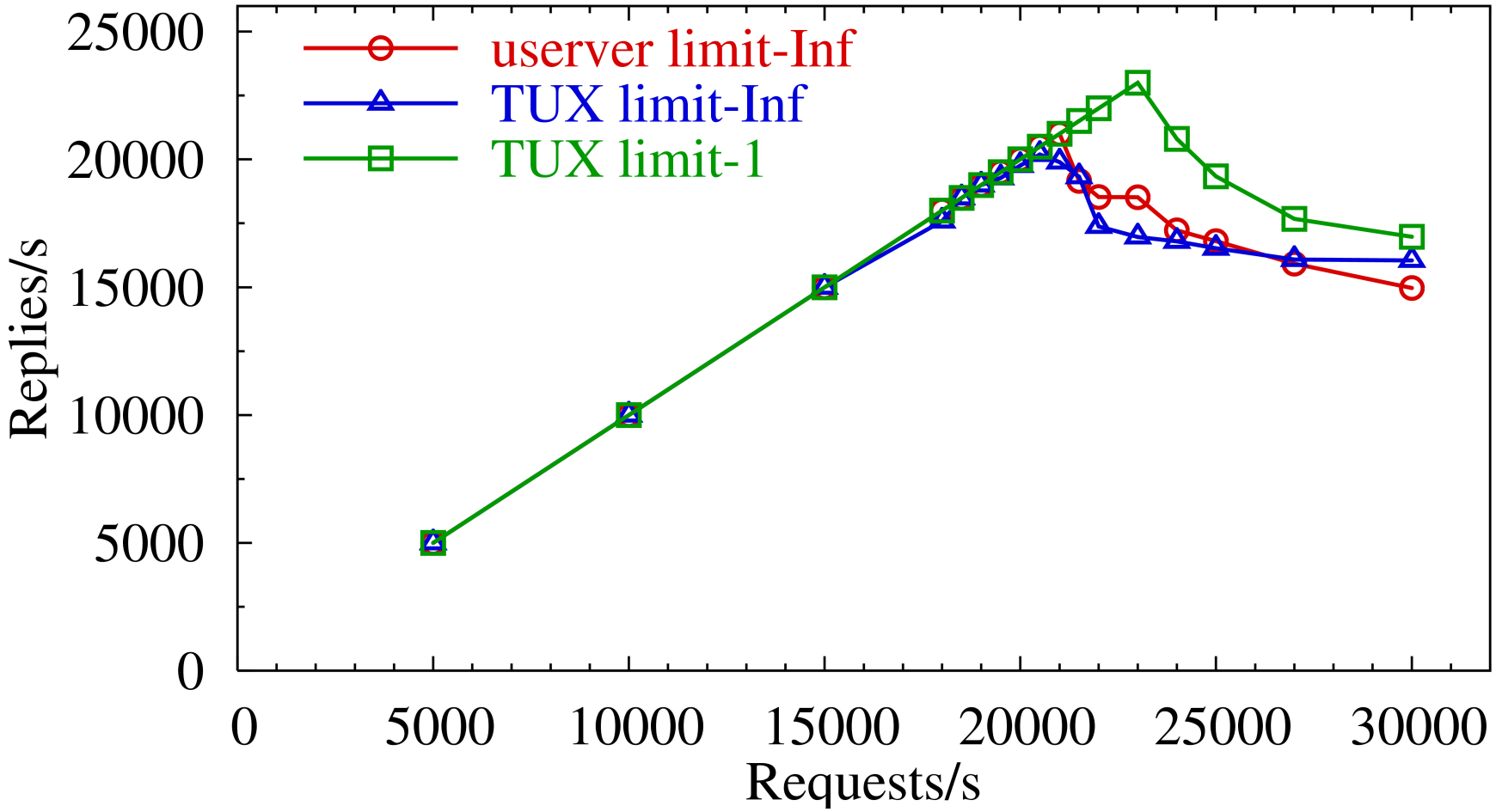
TUX vs the userver (one packet)



TUX vs the userver (one packet)

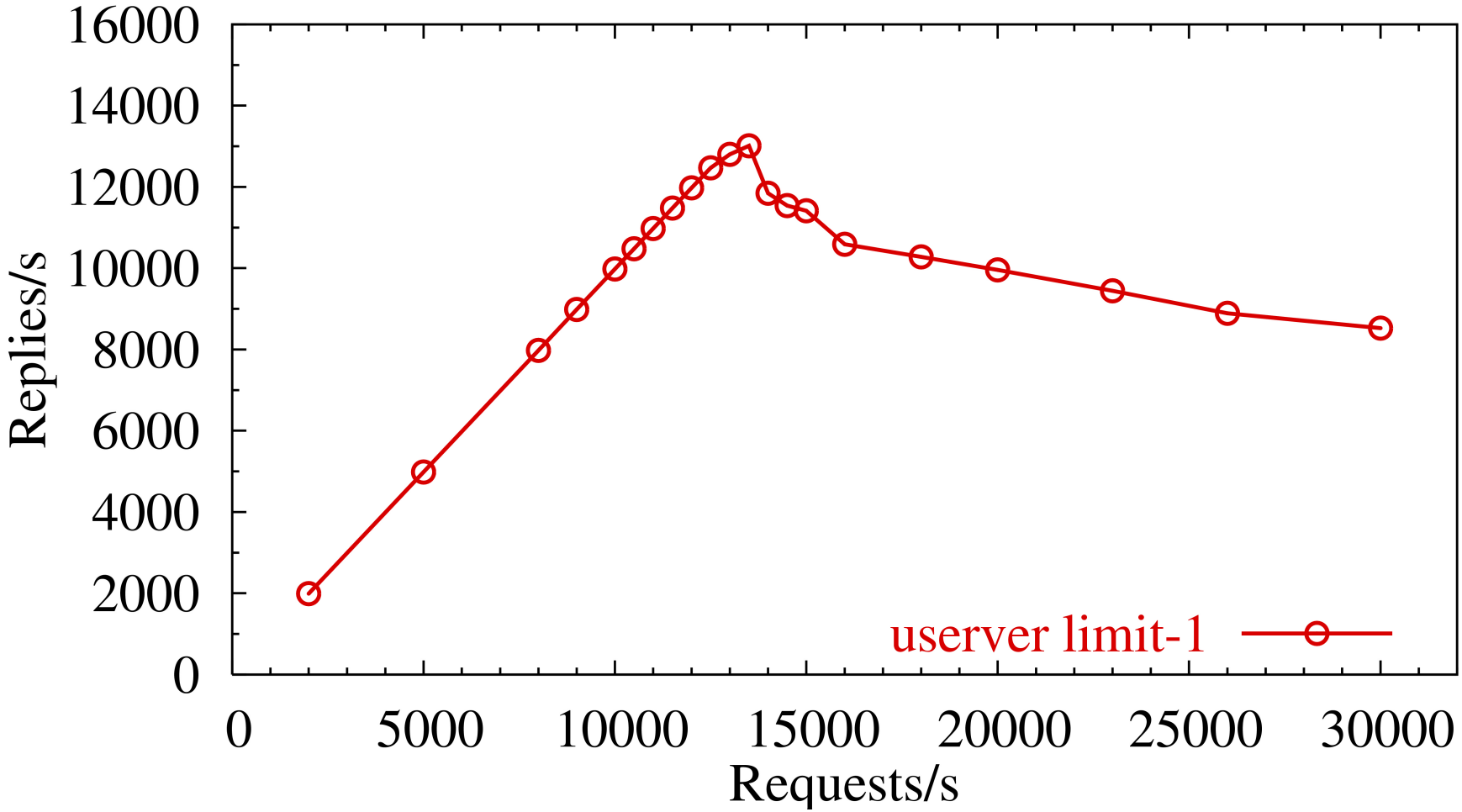


TUX vs the userver (one packet)

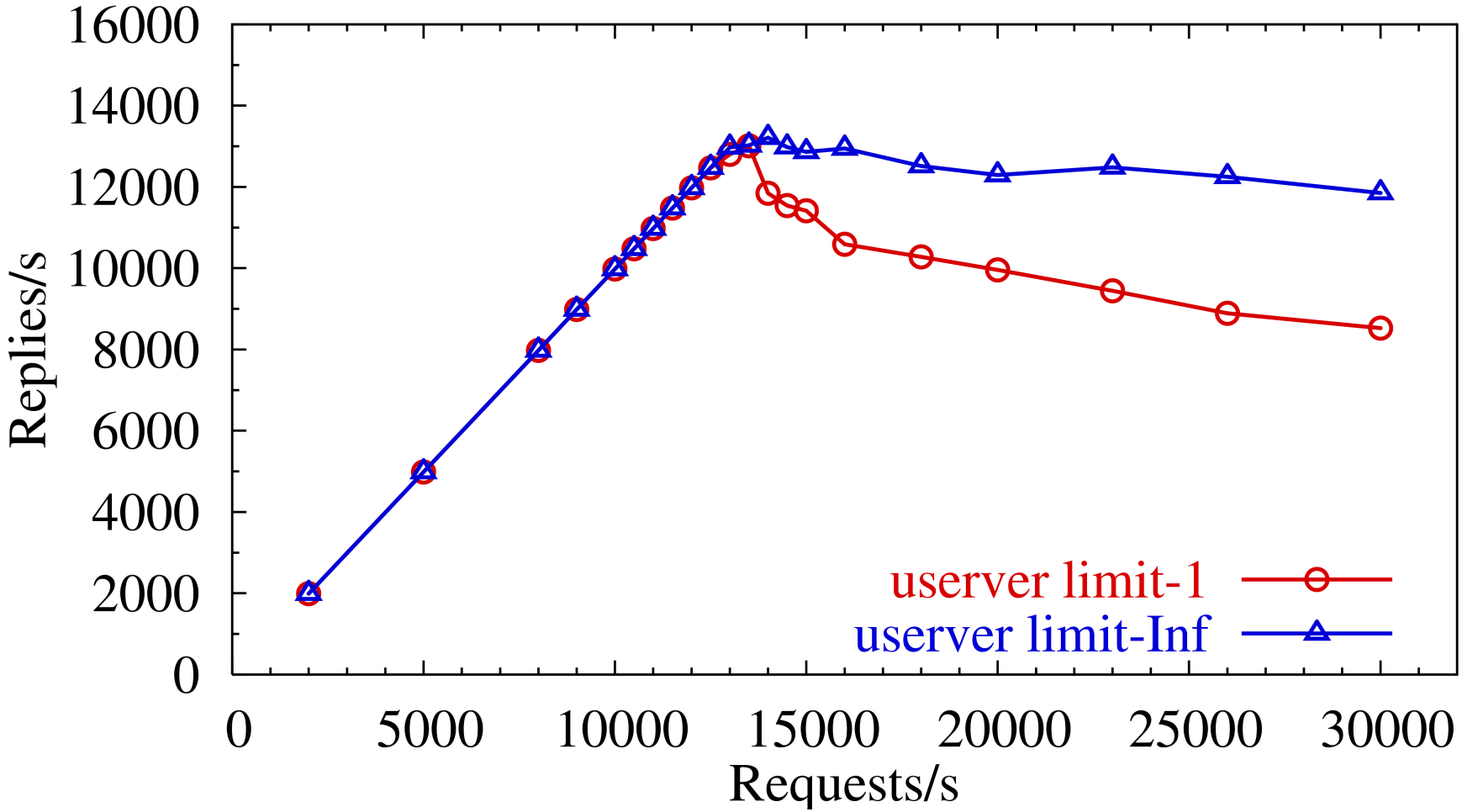


Knot was run on different cluster and hasn't been tuned

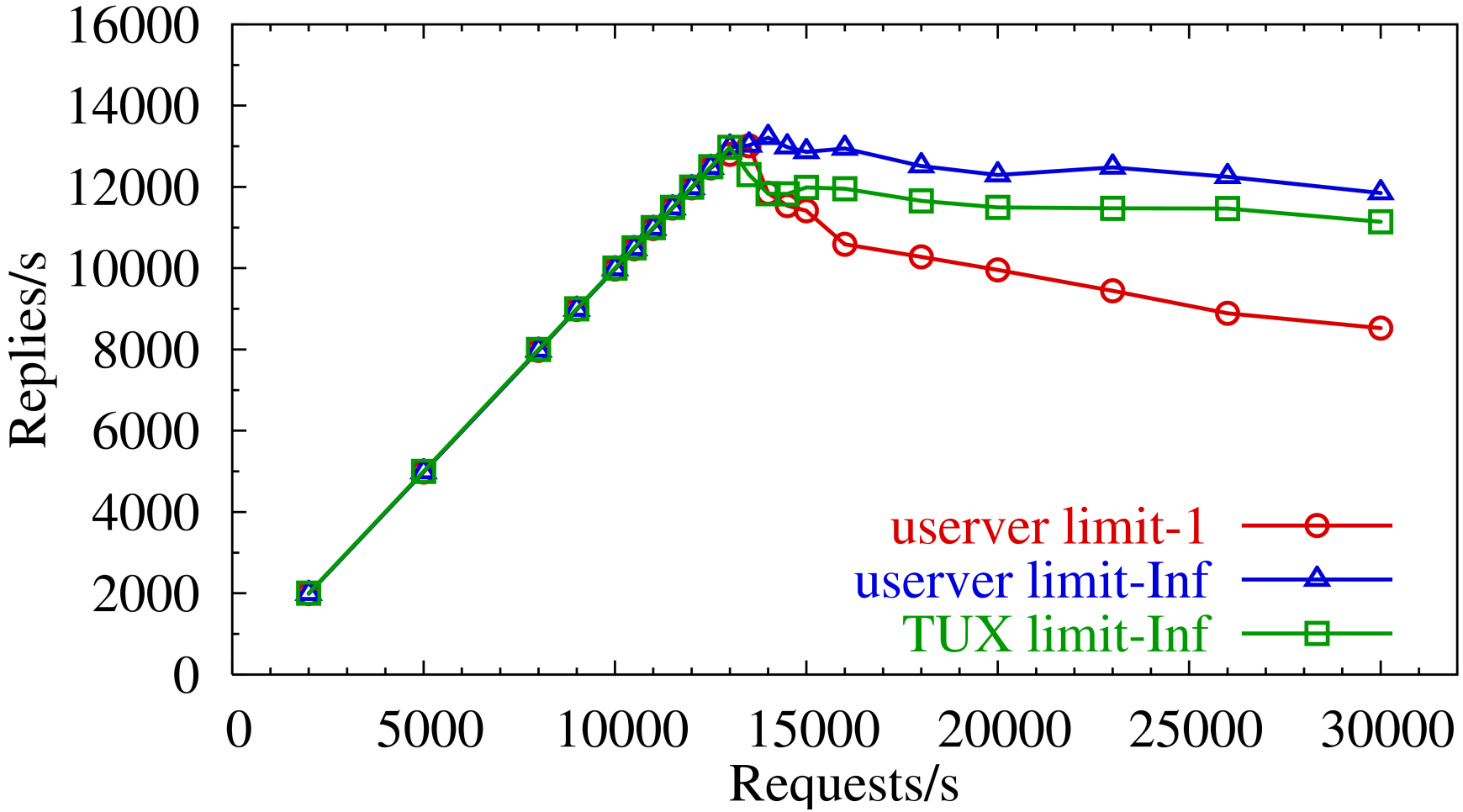
TUX vs the userver (SPECweb99-like)



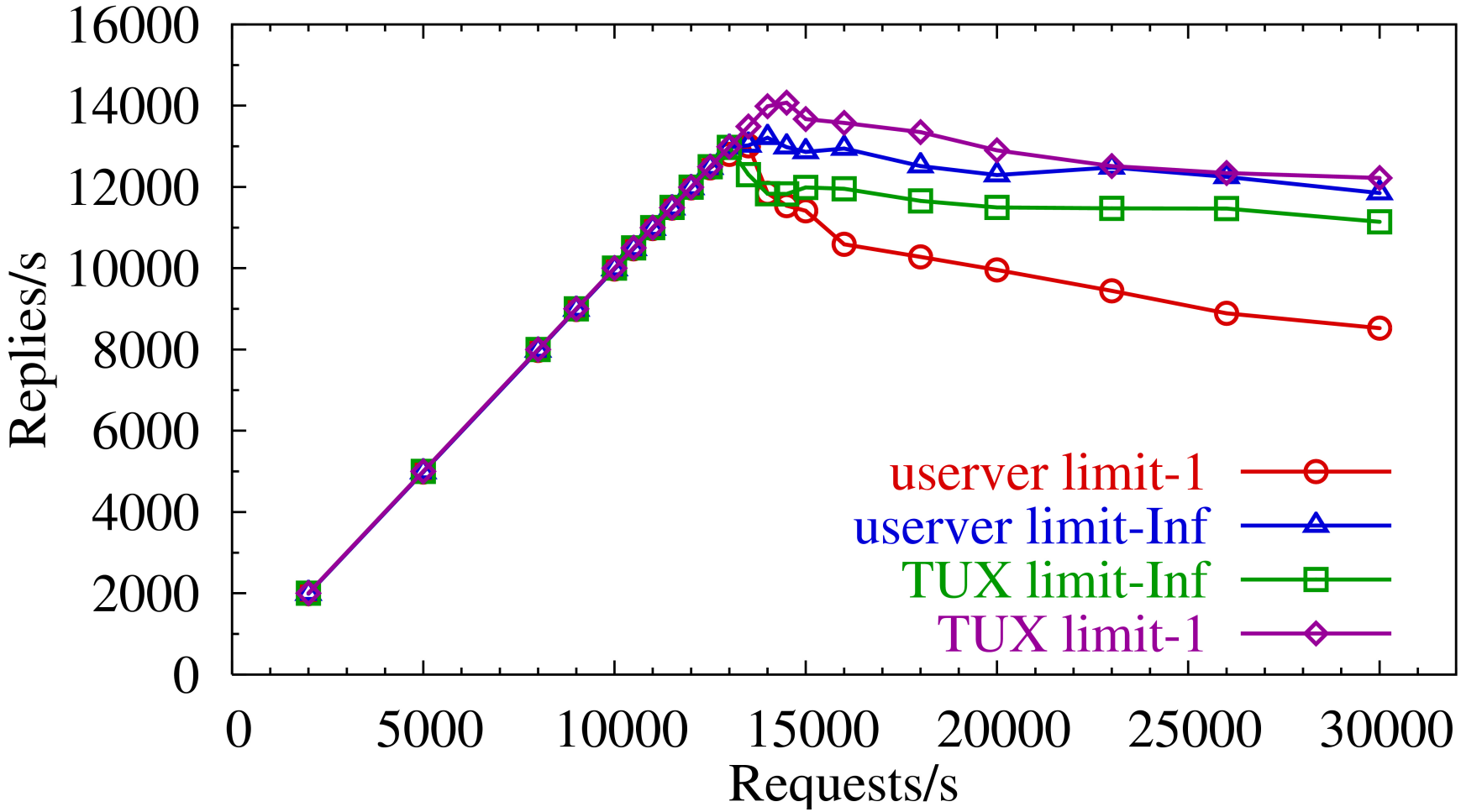
TUX vs the userver (SPECweb99-like)



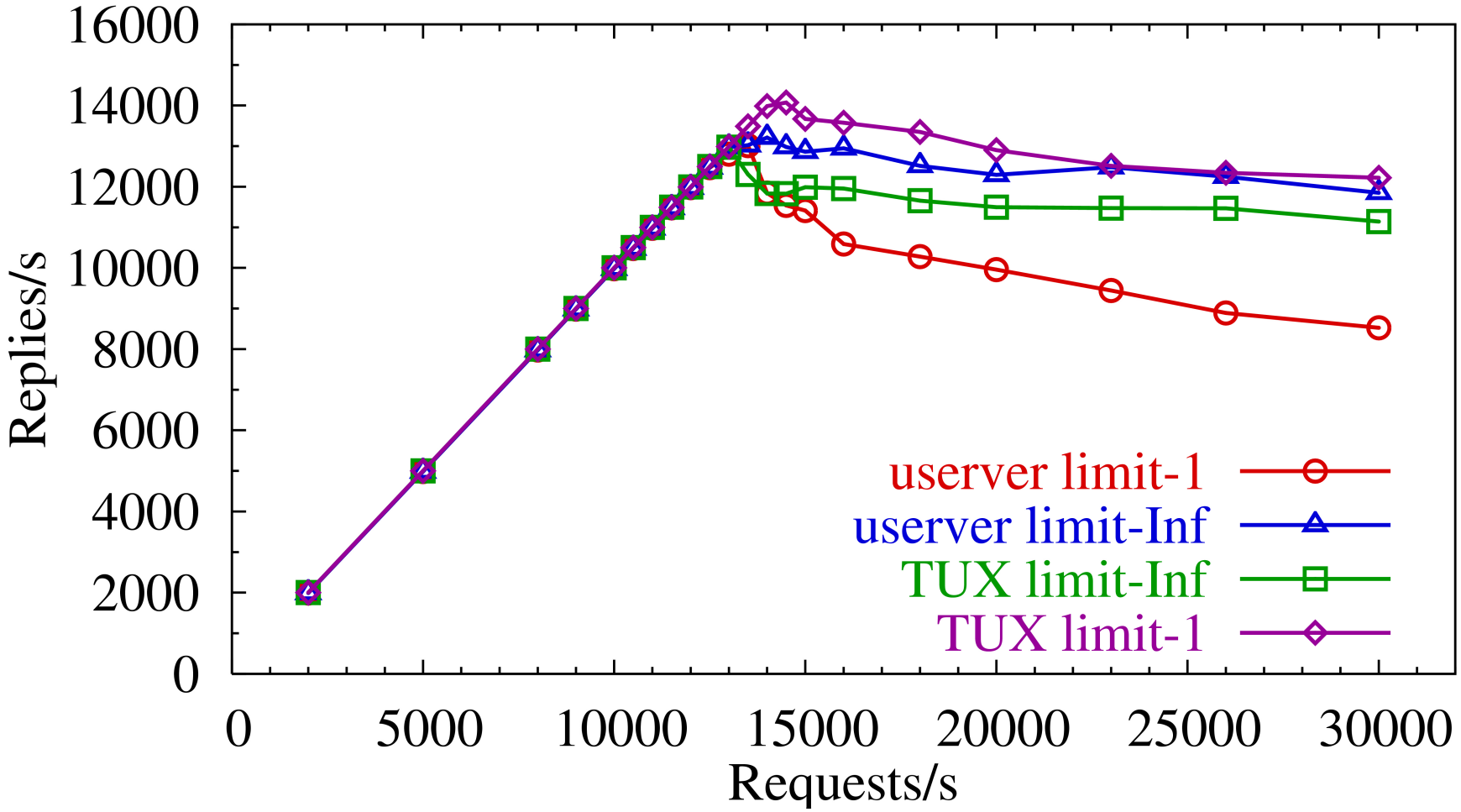
TUX vs the userver (SPECweb99-like)



TUX vs the userver (SPECweb99-like)

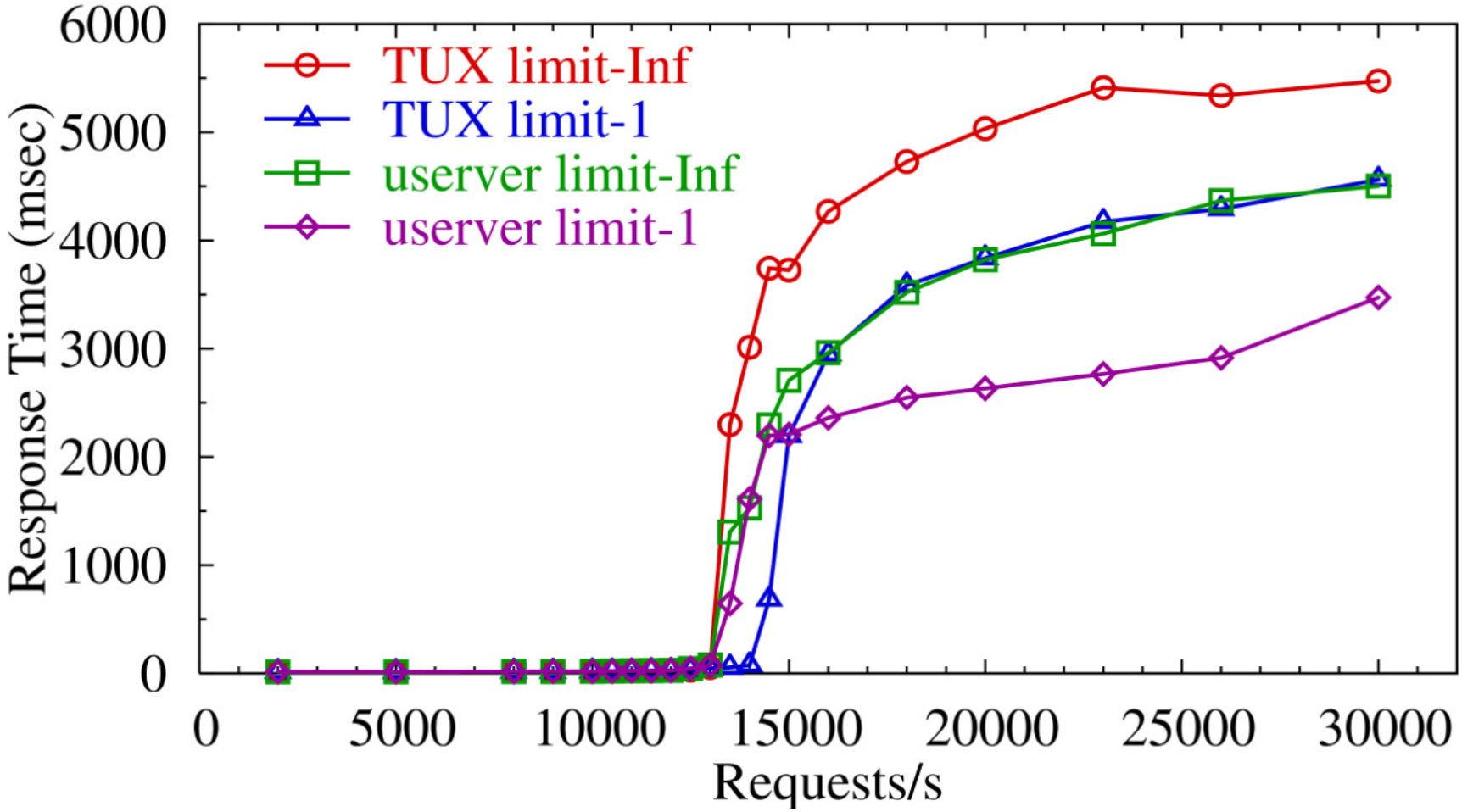


TUX vs the userver (SPECweb99-like)



No improvements for Knot with SPECweb99-like wload

Latencies for SPECweb99-like wload



Conclusions

- Ensure accepting at a sufficiently high rate
- Balance accepting with working on existing conns

- Demonstrated performance improvements for:
  - **`userver`** (event-driven user-mode)
  - **`Knot`** (thread-based user-mode)
  - **`TUX`** (event-driven kernel-mode)
- user-mode close to kernel-mode for these workloads

The End