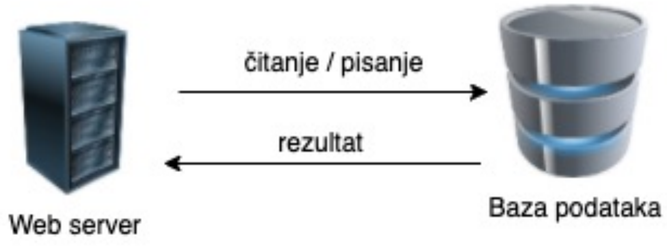
# SERVERSKE ARHITEKTURE

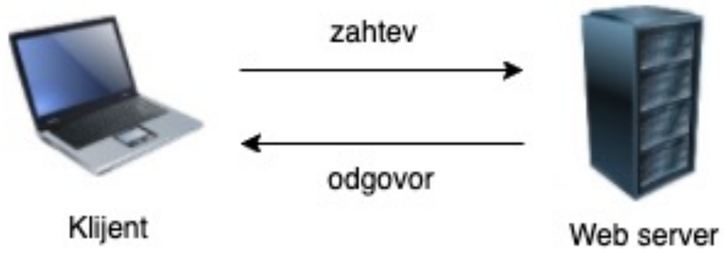
# SCENARIJI TOKOVA PODATAKA

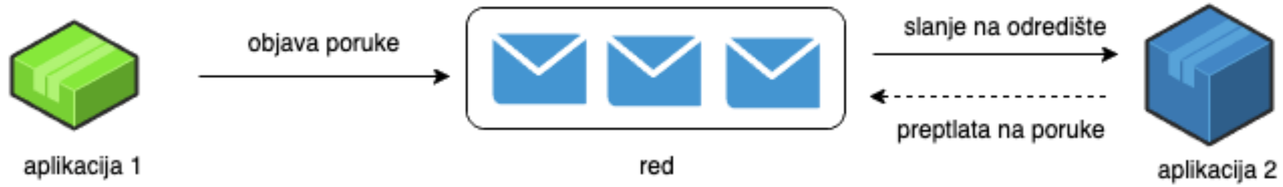


## TRI NAJČEŠĆA SCENARIJA

- Serverska aplikacija <-> Baza podataka
- Direktna komunikacija klijent <-> server kroz poziv servisa
- Asinhrona komunikacija razmenom poruka preko reda poruka (message queue)



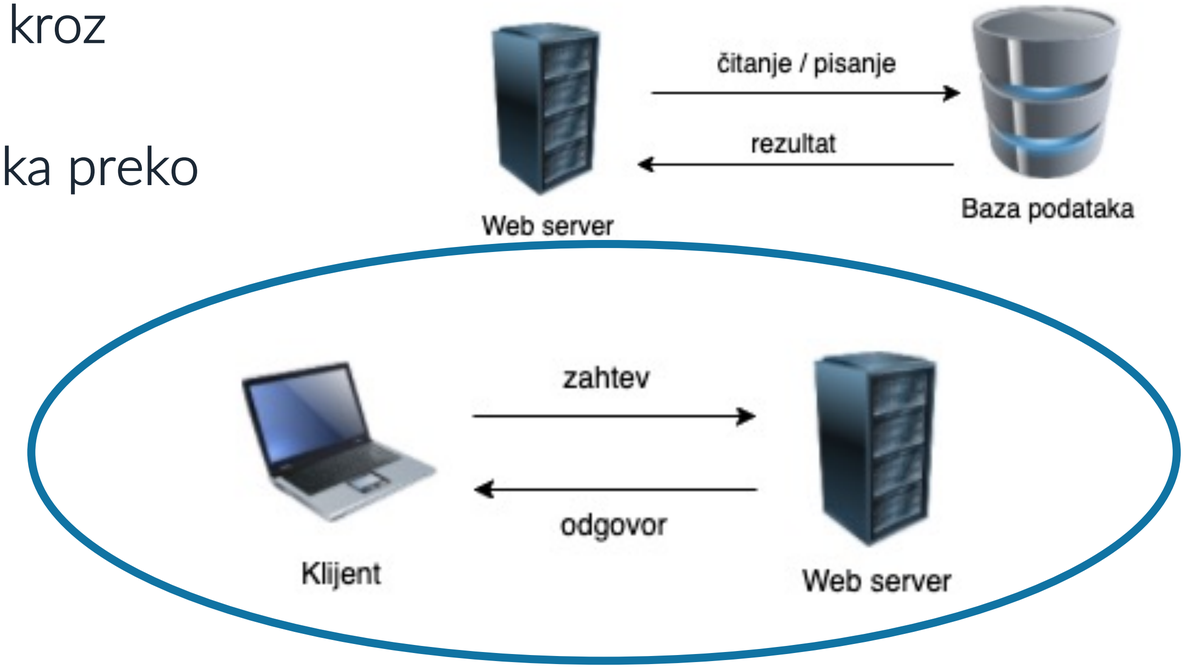



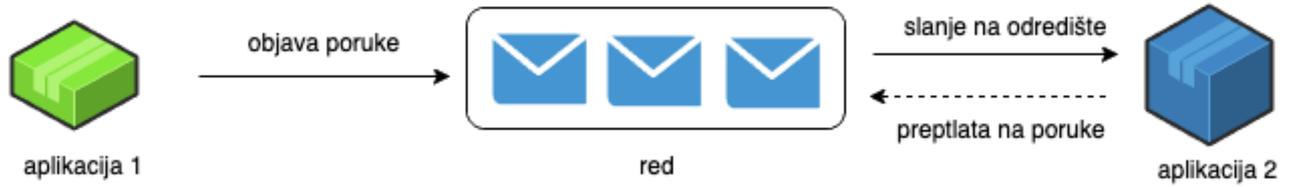
# SCENARIJI TOKOVA PODATAKA



## TRI NAJČEŠĆA SCENARIJA

- Serverska aplikacija <-> Baza podataka
- Direktna komunikacija klijent <-> server kroz poziv servisa
- Asinhrona komunikacija razmenom poruka preko reda poruka (message queue)



# WEB SERVER



## ČEMU SLUŽI?

- Komponenta odgovorna za prihvatanje HTTP zahteva i vraćanje odgovora
- Dizajniran je da služi statički sadržaj (ali mnogi imaju podršku za scripting jezike poput Pearl, PHP, itd. za generisanje dinamičkog HTTP sadržaja)

## POZNATI SERVERI

- Apache HTTP Server
- Nginx
- lighttpd
- ...

# APLIKATIVNI SERVER



## ČEMU SLUŽI?

- Generiše dimanički sadržaj koji dolazi kao rezultat izvršavanja poslovne logike
- Većina aplikativnih servera ima web server kao svoj sastavni deo, te može da radi iste stvari
- Dodatno može imati podršku za *connection pooling*, *messaging* servise, itd.
- Nije limitiran na korišćenje samo HTTP protokola

## POZNATI SERVERI

- Apache Tomcat
- WildFly (JBoss)
- Oracle WebLogic
- Gunicorn
- ...

# KORACI ZA OBRADU ZAHTEVA



## PRIHVATANJE ZAHTEVA

- Ako je u pitanju novi zahtev, prvo mora da se uspostavi HTTP konekcija preko TCP konekcije

## INICIJALNA OBRADA ZAHTEVA

- Podrazumeva čitanje bajtova (*I/O bound*) i parsiranje HTTP zahteva (*CPU bound*)
- Ako je u pitanju POST ili PUT zahtev, zahteva se dodatno procesiranje podataka koji se šalju u zahtevu

## SLANJE ZAHTEVA APLIKACIJI NA DALJU OBRADU

- Podrazumeva slanje zahteva sloju poslovne logike
- Prosleđivanje poziva se može svesti na čitanje podataka sa fajl sistema/baze podataka, slanje zahteva preko mreže na neki message queue, RPC poziv,... (*I/O bound*)

# KORACI ZA OBRADU ZAHTEVA



## KREIRANJE ODGOVORA

- Kada je zahtev obrađen i resurs je spreman (HTML stranica, slika, video, ...), vraća se klijentu pisanjem na socket

## ZAVRŠETAK OBRADE ZAHTEVA

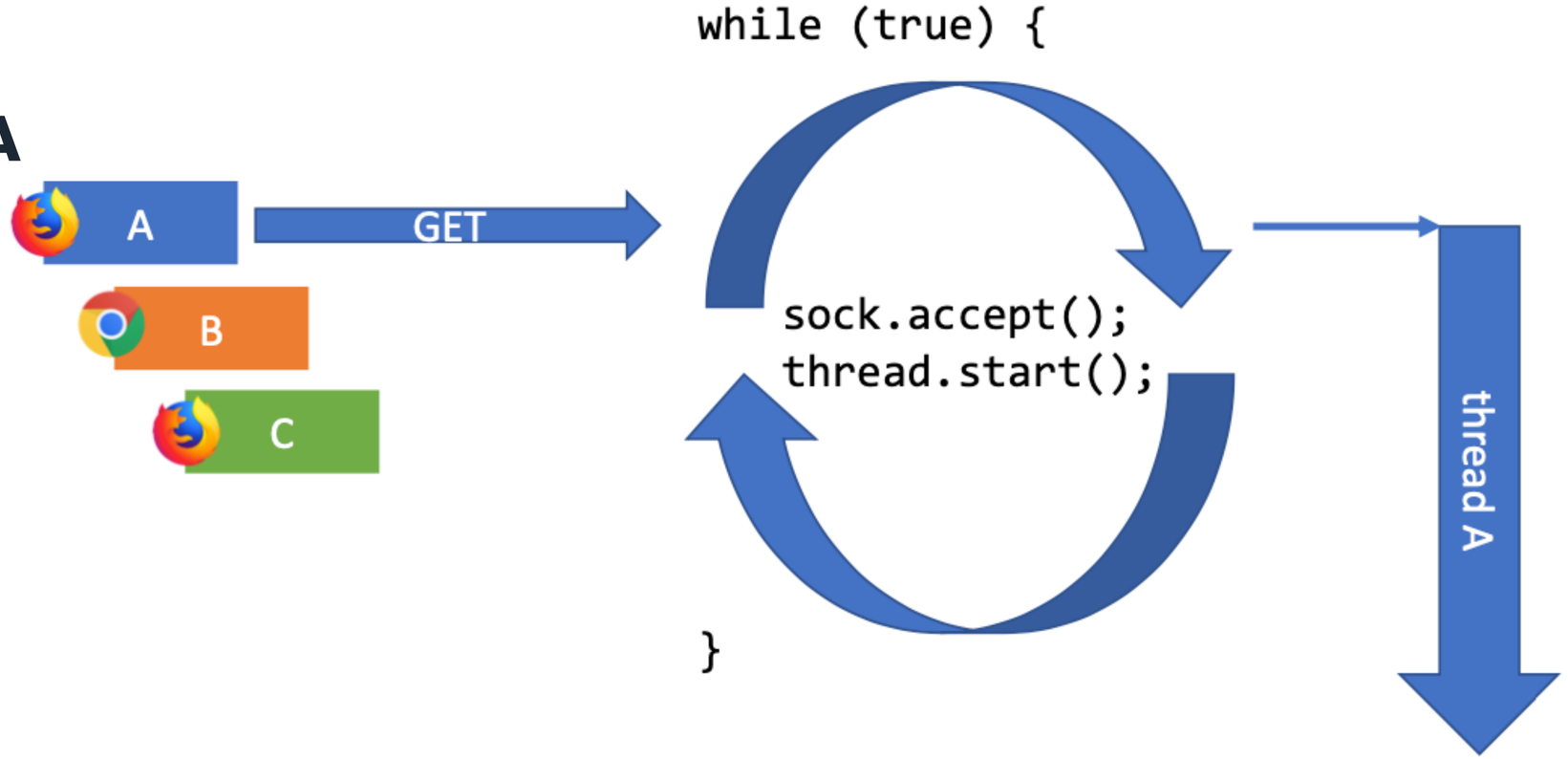
- Web server zatvara konekciju ili se vraća na prvi korak i čeka sledeći zahtev

# SERVERSKE ARHITEKTURE BAZIRANE NA NITIMA



## TOK OBRADE ZAHTEVA

- Klijent A šalje GET zahtev serveru
- Serverska glavna petlja pokreće novu nit za obradu zahteva
- Nit počinje obradu zahteva paralelno sa glavnom petljom
- Glavna petlja ponovo čeka novi zahtev




slika preuzeta sa https://github.com/mbranko/isa19/blob/master/01-threads/thread-models.pdf

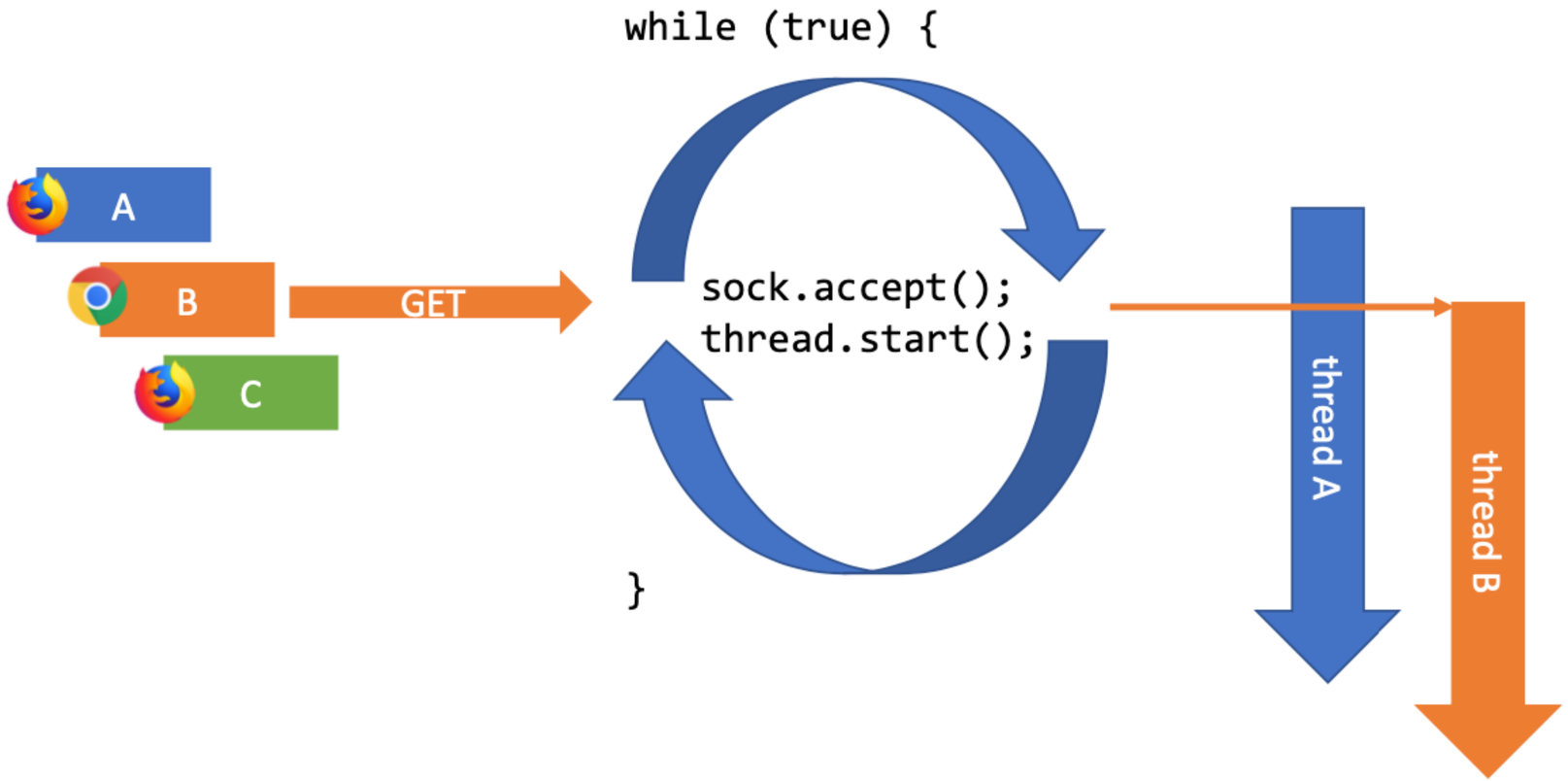
thread-based-server primer

# SERVERSKE ARHITEKTURE BAZIRANE NA NITIMA



## TOK OBRADE ZAHTEVA

- Klijent B šalje GET zahtev serveru
- Serverska glavna petlja pokreće novu nit za obradu zahteva
- Nit počinje obradu zahteva paralelno sa glavnom petljom
- Glavna petlja ponovo čeka novi zahtev
- Prethodna nit za obradu zahteva još nije završila rad




slika preuzeta sa https://github.com/mbranko/isa19/blob/master/01-threads/thread-models.pdf

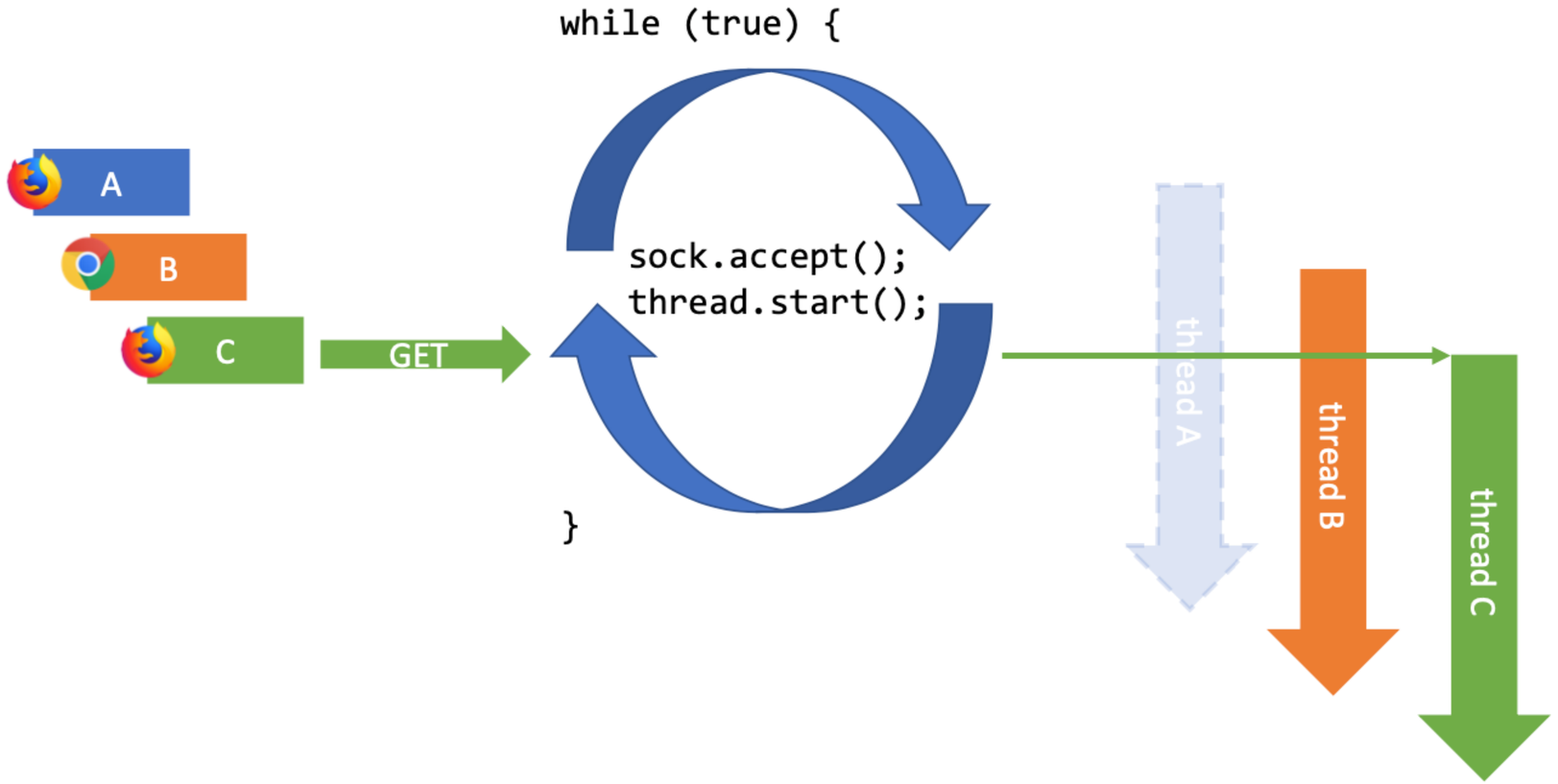
thread-based-server primer

# SERVERSKE ARHITEKTURE BAZIRANE NA NITIMA



## TOK OBRADE ZAHTEVA

- Klijent C šalje GET zahtev serveru
- Serverska glavna petlja pokreće novu nit za obradu zahteva
- Nit počinje obradu zahteva paralelno sa glavnom petljom
- Glavna petlja ponovo čeka novi zahtev
- Prva nit za obradu zahteva je završila rad
- Druga nit još uvek radi

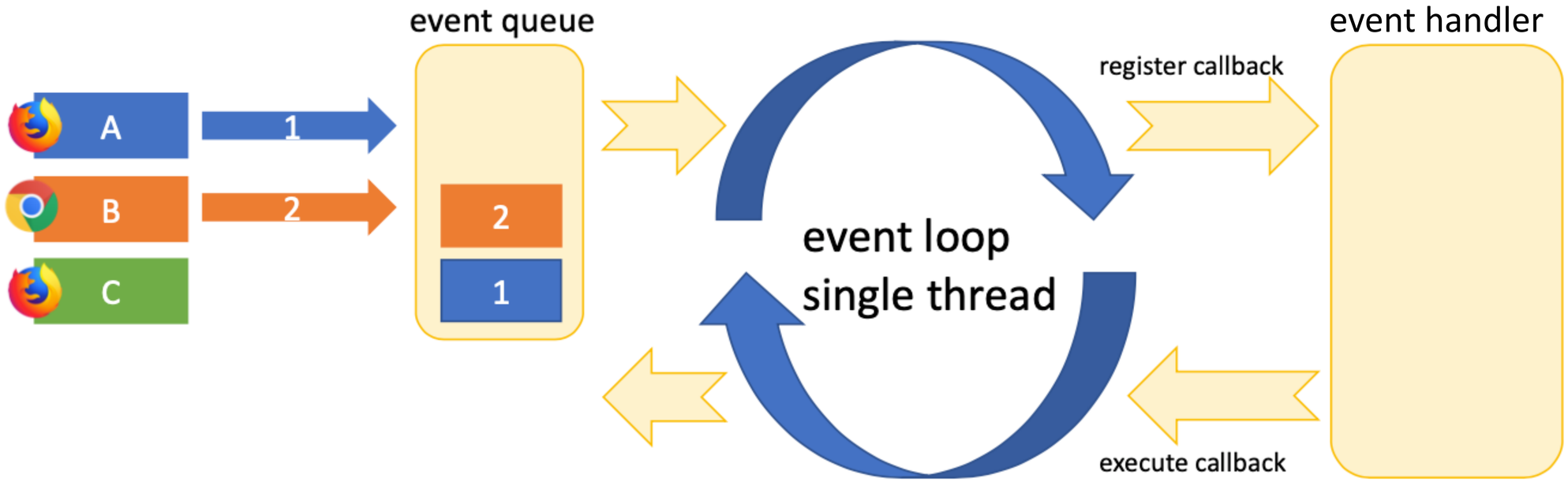



slika preuzeta sa https://github.com/mbranko/isa19/blob/master/01-threads/thread-models.pdf

thread-based-server primer

# SERVERSKE ARHITEKTURE BAZIRANE NA DOGAĐAJIMA



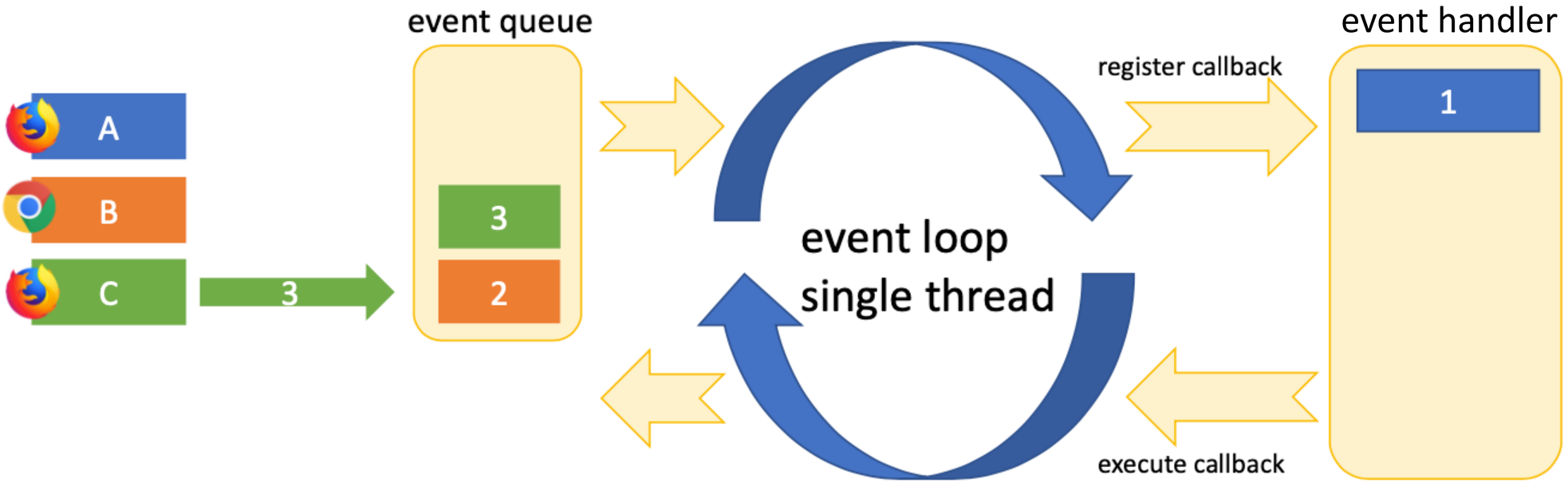



slika preuzeta sa https://github.com/mbranko/isa19/blob/master/01-threads/thread-models.pdf

event-based-server primer

# SERVERSKE ARHITEKTURE BAZIRANE NA DOGAĐAJIMA



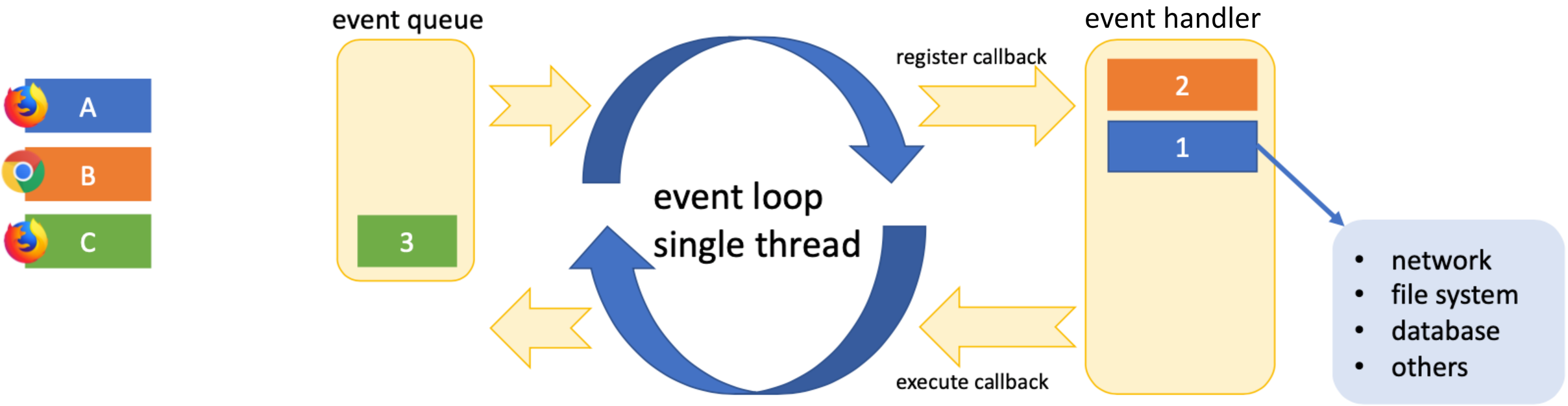



slika preuzeta sa https://github.com/mbranko/isa19/blob/master/01-threads/thread-models.pdf

event-based-server primer

# SERVERSKE ARHITEKTURE BAZIRANE NA DOGAĐAJIMA






slika preuzeta sa https://github.com/mbranko/isa19/blob/master/01-threads/thread-models.pdf

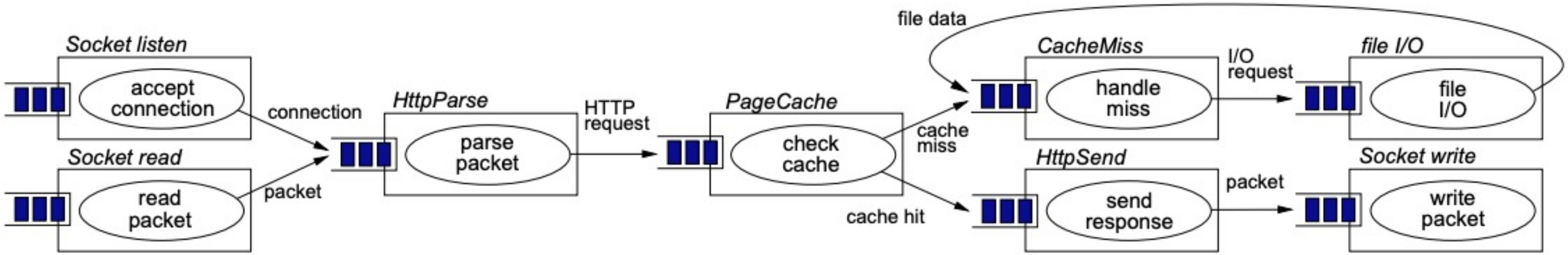
event-based-server primer

# STAGED EVENT-DRIVEN ARCHITECTURE (SEDA)



## KOMBINOVANA ARHITEKTURA KOJA KORISTI I NITI I DOGAĐAJE

- Predložili Matt Welsh, David Culler i Eric Brewer [1] s namerom da unaprede performanse
- U osnovi je podeljena serverska logika u lanac striktno definisanih faza
- Faze su povezane pomoću redova
- Zahtevi se prosleđuju iz jedne u drugu fazu tokom procesiranja
- Svaka faza ima nit ili skup niti (thread pool) koji mogu da se konfigurišu dinamički



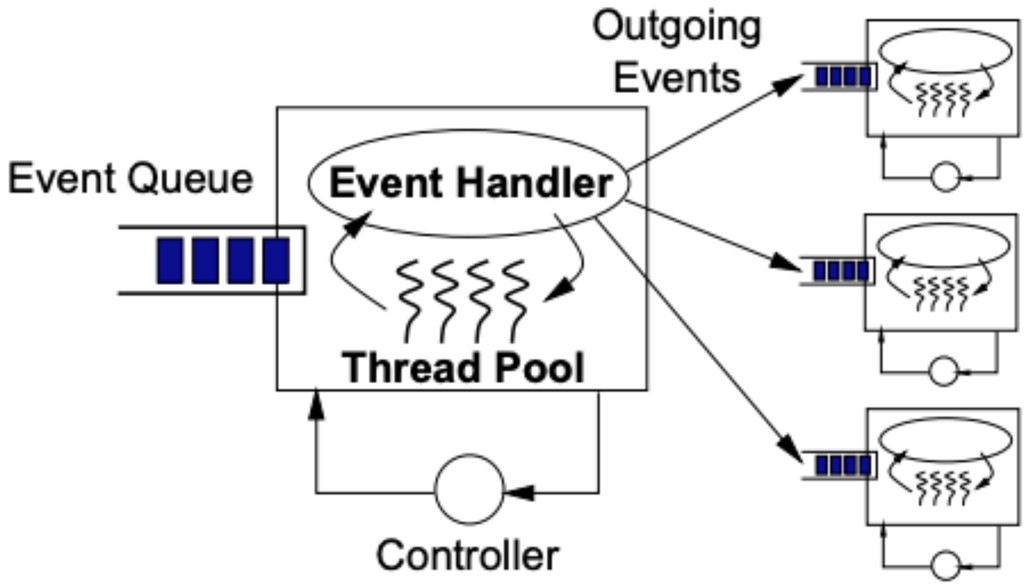

[1] https://docs.huihoo.com/seda/seda-sosp01.pdf

slika preuzeta iz istog rada

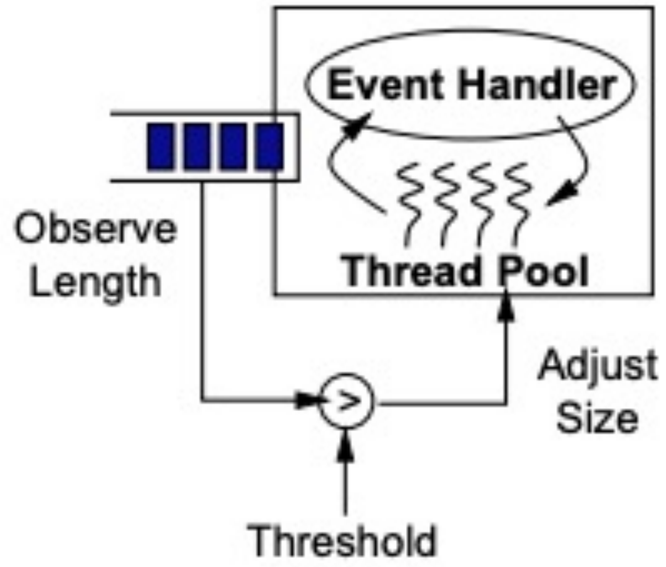
# STAGED EVENT-DRIVEN ARCHITECTURE (SEDA)



## SEDA FAZA




## SEDA KONTROLER




[1] https://docs.huihoo.com/seda/seda-sosp01.pdf

slike preuzeta iz istog rada

# C10K PROBLEM



## KAKO WEB SERVERI MOGU DA IZAĐU NA KRAJ SA 10.000 ISTOVREMENIH ZAHTEVA?

- Dan Kegel 1999. objavio članak [1]
- Danas predstavlja osnovni resurs za diskusiju o skalabilnosti web servera
- Smatra da (u to vreme) hardver ne mora biti usko grlo sistema za obradu konkurentnih konekcija
- Izdvaja strategije poput opsluživanja više klijenata sa jednom niti i korišćenje neblokirajućih I/O operacija na određeni način
- Problem je rešen izmenama u kernelu OS i prelaskom na event-driven servere (npr. Ngnix i Node bazirane)
- Osnova za nove probleme C100K, C1M, C10M,...

[1] http://www.kegel.com/c10k.html

# METRIKE ZA MERENJE PERFORMANSI SERVERA

## METRIKE ZA MERENJE PERFORMANSI SERVERA

- Propusni opseg zahteva (request throughput meren kao broj zahteva po jedinici vremena – req/sec)
- Propusni opseg podataka (meren u Mbps)
- Vreme odgovora (response time meren u ms)
- Broj konkurentnih konekcija (izražen kao broj)

## DODATNE METRIKE UKLJUČUJU MONITORING SERVERSKE MAŠINE

- Zauzeće CPU
- Zauzeće memorije
- Broj niti/procesa
- Broj otvorenih soketa
- Broj otvorenih fajlova
- ...

# REFERENCE




- PRIMERI PO UZORU NA https://github.com/mbranko/isa19/tree/master/01-threads
- PRIMER ZA MERENJE PERFORMANSI https://github.com/mbranko/isa19/tree/master/01-threads/analyze
- WELSH ET AL. AN ARCHITECTURE FOR WELL-CONDITIONED, SCALABLE INTERNET SERVICES. https://docs.huihoo.com/seda/seda-sosp01.pdf
- DAN KAGEL. C10K PROBLEM. http://www.kegel.com/c10k.html
- PARIAG ET AL. COMPARING THE PERFORMANCE OF WEB SERVER ARCHITECTURES. https://people.eecs.berkeley.edu/~brewer/cs262/Pariag07.pdf
- NODE.JS EVENT LOOP. https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/
- BENJAMIN ERB. CONCURRENT PROGRAMMING FOR SCALABLE WEB ARCHITECTURES. https://berb.github.io/diploma-thesis/
- NIKHIL MARATHE. AN INTRODUCTION TO LIBUV. https://nikhilm.github.io/uvbook/basics.html

# KOJA SU VAŠA PITANJA?