Integracija softverskih tehnologija

Kanali za vezu, osobine povezivanja i defnisanje ponašanja servisa

Zoran Ćirović

Uvod

- Kanal je provodnik kroz koji prolaze poruke ka i od WCF servisa. Aplikacija je odgovorna za pripremu i isporuku poruke na dosledan način. Kanali su definisani za prenos, protokole i presretanje poruka.
- Kanal se sastoji od slojeva koji zajedno čine kanalski stek. Na primer, kanalski stek se može sastojati od TCP transportnog kanala i od transakcionog protokolskog kanala.
- Kanalski stek transformiše poruke u žični format kompatibilan za pošiljaoca i prijemnika poruke.

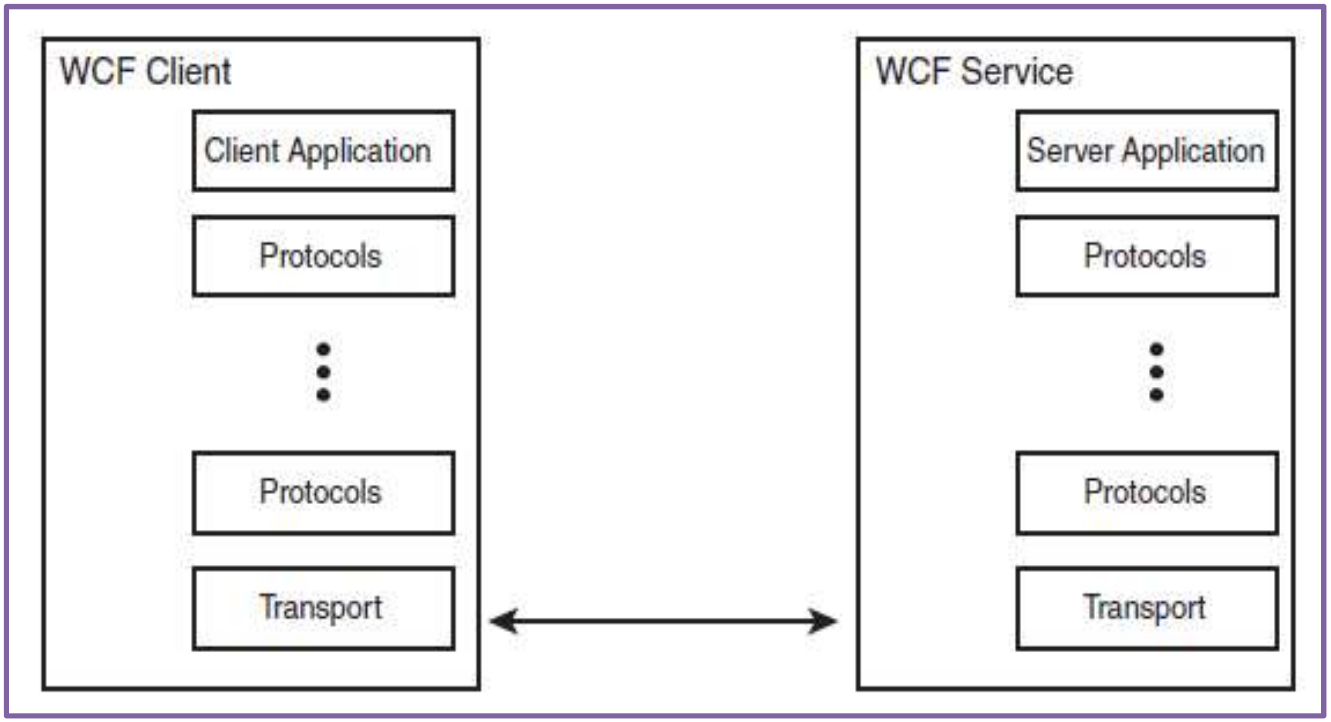
- Postoje dva tipa kanala koji se koriste:
  - Transportni kanal
  - Protokolni kanal

Transportni / protokolni kanal

- **Transportni kanal** je uvek na dnu kanalskog steka i odgovoran je za prenos poruka pomoću transportnog protokola. WCF omogućava upotrebu nekoliko transportnih protokola:
  - HTTP, TCP, MSMQ, peer-to-peer, named pipes
- **Protokolni kanali** su iznad transportnih. Omogućavaju realizaciju protokola žičnog nivoa transformišući poruke. WCF podržava puno tipova protokolnih kanala:
  - Za implementaciju sigurnosti, transakcija i pouzdanosti poruka
- Klijent i server implementiraju kompatibilne kanalne stekove. Da bi olakšali kreiranje kanala koristi se svojstvo povezivanja - ***binding***. Svojstvo povezivanja ima konfiguraciju kanalnog steka i ume da ga kreira pri kompajliranju. Povezivanja su sastavljena od kolekcije ***binding elemenata***, koji tipično predstavljaju kanale u kanalnom steku.

Arhitektura kanala

- Arhitektura WCF kanala poseduje veliku fleksibilnost, zahvaljujući, između ostalog, apstrakovanju komunikacije.



Povezivanja

Osobine i primena

Definicija

- Povezivanja *(engl. bindings)* su prekonfigurisani kanalski stekovi.
- Drugim rečima, **predstavljaju dogovor na žičnom** (neposrednom, prenosnom) nivou između klijenta i servisa. Svako povezivanje tačno određuje: transport, kodovanje i protokole koji učestvuju u toj komunikaciji.
- Primer:
  - *basicHttpBinding* je namenjen za rad sa servisima zasnovanim na ASP.NET Web servisima ili WS-I Basic Profile 1.1 komunikacionim servisima.
  - *ws2007HttpBinding* i *wsHttpBinding* su slični sa basicHttpBinding, ali podržavaju više opcija kao što je WS-Addressing. ws2007HttpBinding je došao sa .NET 3.5 i zasnovan je na novijim standardima nego wsHttpBinding.

Povezivanje se može definisati u konfiguracionom fajlu ili u kodu:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.serviceModel>
  <services>
   <service name="WCFconf.Cenovnik">
    <host>
     <baseAddresses>
      <add baseAddress="http://localhost:8000/cenovnik"/>
     </baseAddresses>
    </host>
    <endpoint address="dodatak" binding="basicHttpBinding"  contract="WCFconf.ICenovnik" />
   </service>
  </services>
 </system.serviceModel>
</configuration>
```

```
ChannelFactory<ICenovnik> cf = new ChannelFactory<ICenovnik>(new  BasicHttpBinding(),
                                new EndpointAddress("http://localhost:8000/Cenovnik/dodatak"));
ICenovnik o = cf.CreateChannel();
string p = o.dajCenu("kafa");
Console.WriteLine("Dobijeno: {0} \n\n", p);
cf.Close();
```

Primer:

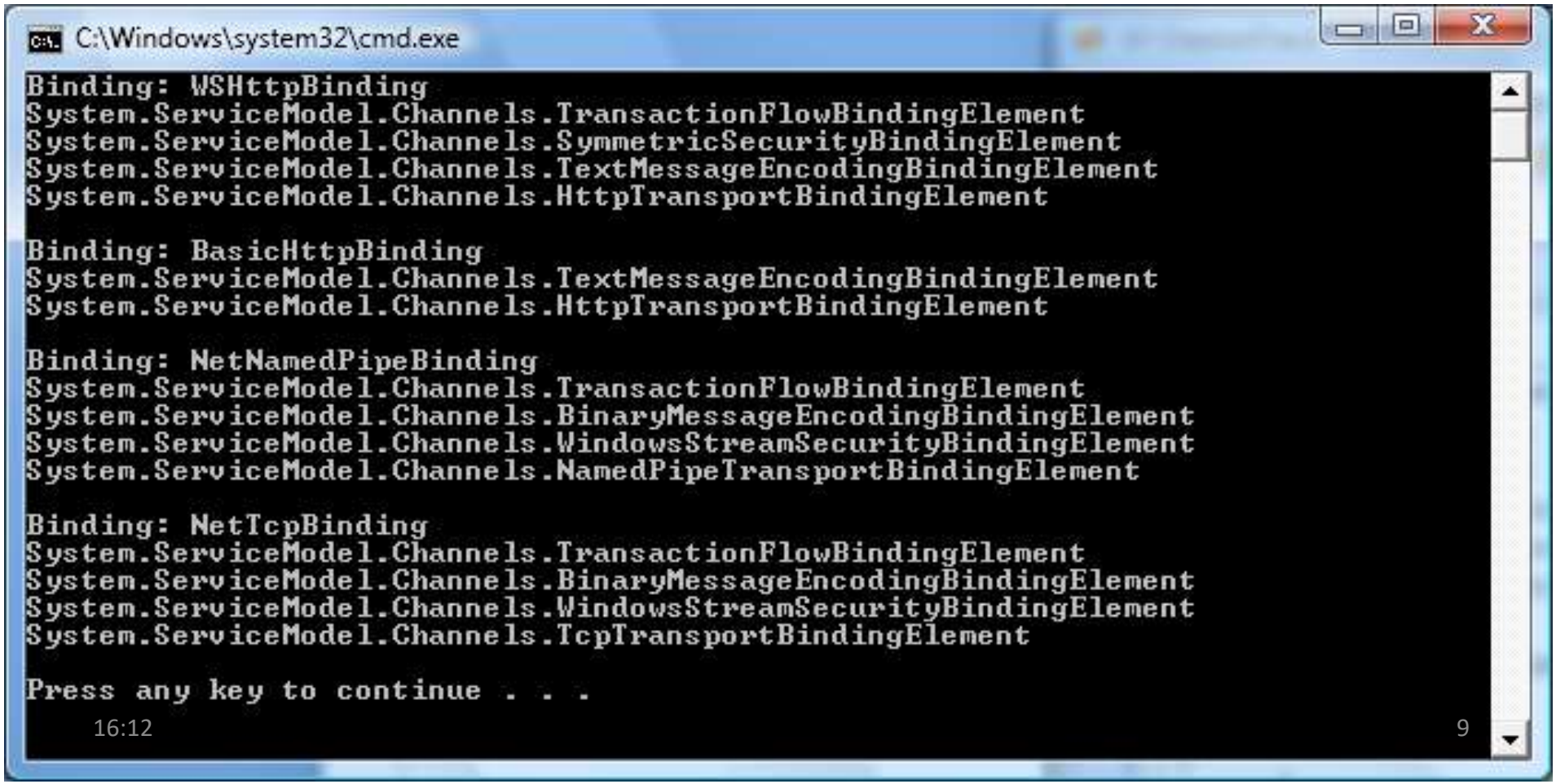
- Napisati primer koji pokazuje sastav određenih predefinisanih povezivanja:
  - wsHttpBinding,
  - NetTcpBinding,
  - NetNamePipeBinding i
  - basicHttpBinding.

- Rešenje:

```
static void Main(string[] args){
        showBindingElements(new WSHttpBinding());
        showBindingElements(new BasicHttpBinding());
        showBindingElements(new NetNamedPipeBinding());
        showBindingElements(new NetTcpBinding());
}
```

```
    static void showBindingElements(Binding b){
        Console.WriteLine("Binding: " + b.Name);
        BindingElementCollection e = b.CreateBindingElements();
        foreach (BindingElement e1 in e){
            Console.WriteLine(e1.GetType().FullName);
        }
        Console.WriteLine();
    }
```

- Rešenje:


C:\Windows\system32\cmd.exe
Binding: WSHttpBinding
System.ServiceModel.Channels.TransactionFlowBindingElement
System.ServiceModel.Channels.SymmetricSecurityBindingElement
System.ServiceModel.Channels.TextMessageEncodingBindingElement
System.ServiceModel.Channels.HttpTransportBindingElement
Binding: BasicHttpBinding
System.ServiceModel.Channels.TextMessageEncodingBindingElement
System.ServiceModel.Channels.HttpTransportBindingElement
Binding: NetNamedPipeBinding
System.ServiceModel.Channels.TransactionFlowBindingElement
System.ServiceModel.Channels.BinaryMessageEncodingBindingElement
System.ServiceModel.Channels.WindowsStreamSecurityBindingElement
System.ServiceModel.Channels.NamedPipeTransportBindingElement
Binding: NetTcpBinding
System.ServiceModel.Channels.TransactionFlowBindingElement
System.ServiceModel.Channels.BinaryMessageEncodingBindingElement
System.ServiceModel.Channels.WindowsStreamSecurityBindingElement
System.ServiceModel.Channels.TcpTransportBindingElement
Press any key to continue . . .

❖ Tumačenje za WSHttpBinding:

- Podrazumevani elementi za WSHttpBinding su: **HttpTransportBindingElement**, **TextMessageEncodingBindingElement**, **SymmetricSecurityBindingElement**, i **TransactionFlowBindingElement.**
- Ovi elementi omogućavaju komunikaciju
  - preko HTTP protokola,
  - *text-base* kodovanje poruka,
  - sigurnost, i
  - transakciju.
- Svi ovi elementi povezivanja čine podrazumevanu konfiguraciju. Moguće je dodati ili oduzeti neki binding element po želji.
- Treba zapaziti da je svako povezivanje sastavljeno od jednog ili više elemenata i da se neki od tih elemenata sreću u raznim drugim povezivanjima. Na primer, wsHttpBinding i basicHttpBinding koriste HttpTransportBindingElement i TextMessageEncodingBindingElement. Ova dva binding-a koriste isti transportni mehanizam, ali se razlikuju po funkcionalnostima i mogućnostima koje pružaju.

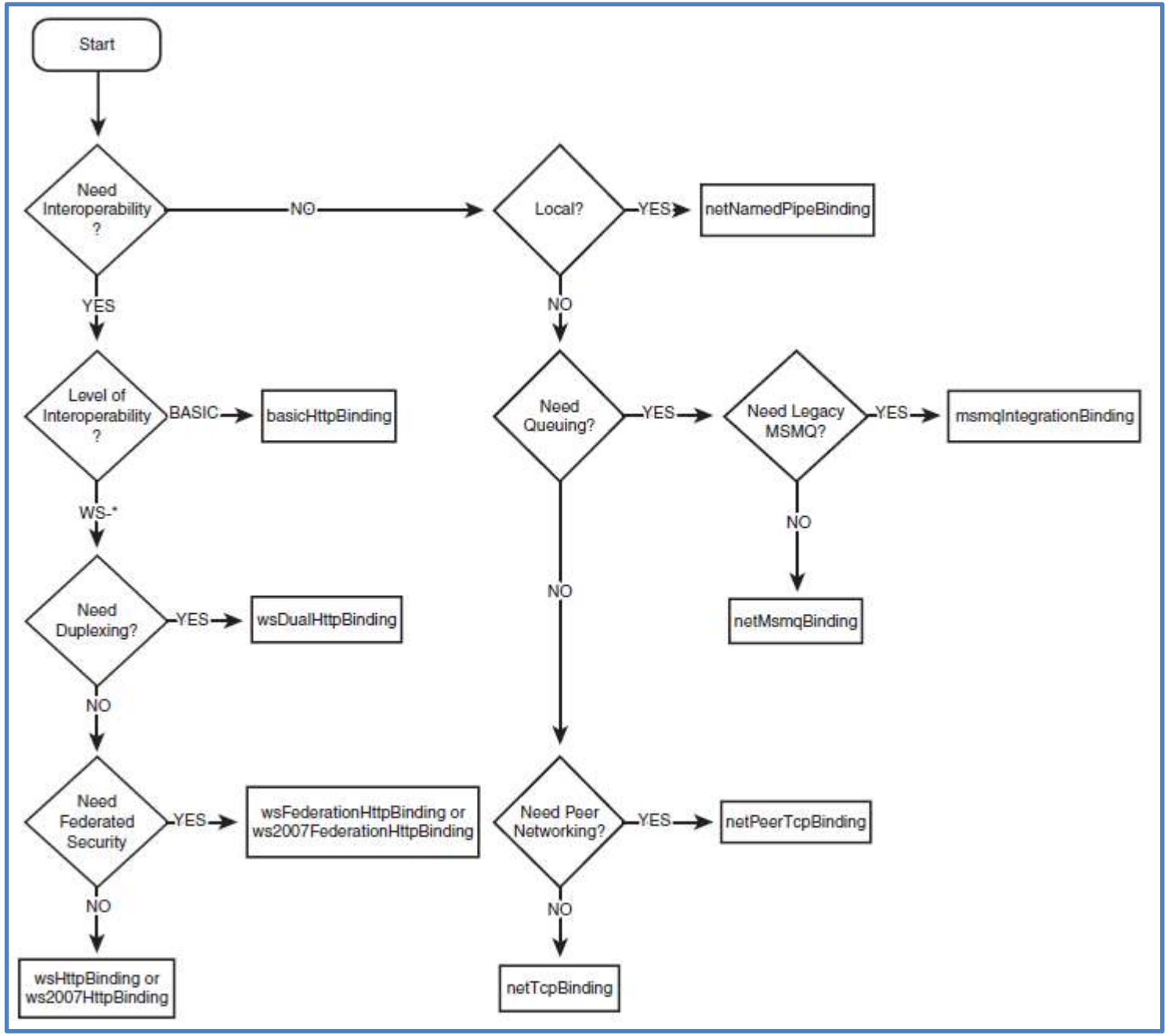
Izbor povezivanja

- Postoji 9 prekonfigurisanih povezivanja u WCFu. Takođe, postoji nekoliko razloga koji određuju koje povezivanje treba odabrati za specifičnu aplikaciju, uključujući sigurnost, interoperabilnost, pouzdanost, performanse i transakcione zahteve. U tabeli, na sledećem slajdu, poredi se 9 povezivanja.

- Na primer, ako aplikacija treba da koristi nepouzdane mreže, kao što je bežična veza, verovatno ćete trebati da obezbedite podršku pouzdanim sesijama – engl. *reliable sessions (RS)*.

Binding Name	Transport-Level Security	Message-Level Security	WS-* Interoperability	WS-*Transactions	Durable Reliable Messaging	Reliable Sessions	Performance	Communication		
								Request/Reply	One-way	Duplex
basicHttpBinding	X	X	X				Good	X	X	
wsHttpBinding	X	X	X	X		RS*	Good	X	X	
wsDualHttpBinding	X	X	X	X		RS*	Good	X	X	X
netTcpBinding	X	X		X		RS*	Better	X	X	X
netNamedPipeBinding	X			X			Best	X	X	X
netMsmqBinding	X	X			X		Better		X	
netPeerTcpBinding	X						Good		X	X
msmqIntegrationBinding	X				X		Better		X	
wsFederationHttpBinding	X	X	X			RS*	Good	X	X	
ws2007HttpBinding	X	X	X	X		RS*	Good	X	X	
ws2007FederationHttpBinding	X	X	X			RS*	Good	X	X	

* RS = WCF Reliable Sessions is an implementation of SOAP reliable messaging defined by the WS-Reliable Messaging (WS-RM) standard.


Start
Need Interoperability ?
NO
Local?
YES
netNamedPipeBinding
YES
NO
Level of Interoperability ?
BASIC
basicHttpBinding
Need Queuing?
YES
Need Legacy MSMQ?
YES
msmqIntegrationBinding
WS-*
NO
Need Duplexing?
YES
wsDualHttpBinding
netMsmqBinding
NO
NO
Need Federated Security
YES
wsFederationHttpBinding or ws2007FederationHttpBinding
Need Peer Networking?
YES
netPeerTcpBinding
NO
NO
wsHttpBinding or ws2007HttpBinding
netTcpBinding

Interoperabilnost

- U nazivu vrste povezivanja postoje prefiksi specifični za vrstu povezivanja.
  Na primer, kada je u pitanju prefiks "**net**" to znači da je takvo povezivanje namenjeno za povezivanje .NET aplikacija.
  Prefiks je indikator koji nam govori koje osobine ima određeno povezivanje i za koje okolnosti je namenjeno.
  To znači da određeno povezivanje poseduje specijalne opcije koje mogu iskoristiti samo .NET aplikacije.
  Suprotno, svako povezivanje koji počinje sa prefiksom "**ws**" je namenjeno za korišćenje sa aplikacijama koje ne moraju da podržavaju .NET.

netTcpBinding

- **netTcpBinding** je dizajniran da podrži komunikaciju između .NET aplikacija koje se nalaze na različitim mašinama povezani mrežom, uključujući i Intranet i Internet. Ovakav oblik komunikacije nazivamo ***cross-machine*** komunikacijom. U ovakvom scenarijumu se ne zahteva interoperabilnost zato što su obe aplikacije bazirane na .NET platformi. Ovo daje veliku fleksibilnost kada je reč o komunikaciji preko mreže, jer je moguća optimizacija kako bi se dobile najbolje preformanse.
- netTcpBinding koristi binarno kodovanje i TCP protokol kako bi postigao najbolje preformanse. Na primer, netTcpBinding nije poželjno koristiti kada *firewall* razdvaja dve .NET aplikacije koje komuniciraju. Najčešće, jedini mogući način komunikacije preko firewall-a je korišćenjem HTTP protokola. Format adresiranja za netTcpBinding je:
- `net.tcp://{hostname}[:port]/{service location}`
- Podrazumevani port za TCP je 808.
- Takođe, po početnim podešavanjima port „*shareing* „je isključen, što bi moglo da ima velikog uticaja ukoliko želimo da otvorimo servis za više aplikacija preko istog porta. Važno svojstvo je *maxConnections* za endpoint, čija je početna vrednost 10.

Osobine netTcpBinding - 1

Attribute Name	Description	Default
`closeTimeout`	The maximum time to wait for the connection to be closed.	00:01:00
`hostNameComparisonMode`	Specifies the method for hostname comparison when parsing URIs.	StrongWildCard
`listenBacklog`	The maximum number of channels waiting to service a request. Any connections greater than this amount are queued.	10
`maxBufferPoolSize`	Maximum size of any buffer pools used by the transport.	524,888
`maxBufferSize`	Maximum number of bytes used to buffer incoming messages in memory.	65,536
`maxConnections`	The maximum number of outbound or inbound connections. Outbound and inbound connections are counted separately.	10
`maxReceivedMessageSize`	The maximum size of an incoming message.	65,536

Osobine netTcpBinding - 2

Attribute Name	Description	Default
`name`	The name of the binding.	n/a
`openTimeout`	The maximum time to wait for an open connection operation to complete.	00:01:00
`portSharingEnabled`	Enable port sharing for the service listener.	false
`readerQuotas`	Specify the complexity of messages that can be processed (for example, size).	n/a
`receiveTimeout`	The maximum time to wait for a receive operation to complete.	00:01:00
`reliableSession`	Specify whether the binding supports exactly once delivery assurances using WS-Reliable Messaging.	n/a
`security`	Specifies the security settings of the binding.	n/a
`sendTimeout`	The maximum time to wait for a send operation to complete.	00:01:00
`transactionFlow`	Enable transactions to flow from the client to the server.	false
`transactionProtocol`	The type of transactions supported—either OleTransactions or WSAtomic-Transactions.	Ole Trans-actions

Konfigurisanje servera za netTcpBinding

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service name="EssentialWCF.StockQuoteService">
        <host>
          <baseAddresses>
            <add baseAddress="net.tcp://localhost/stockquoteservice" />
          </baseAddresses>
        </host>
        <endpoint address=""
                  contract="EssentialWCF.IStockQuoteService"
                  binding="netTcpBinding" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Konfigurisanje klijenta za netTcpBinding

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <client>
      <endpoint address="net.tcp://localhost/stockquoteservice"
                binding="netTcpBinding"
                contract="EssentialWCF.IStockQuoteService">
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

Komunikacija .NET aplikacija na istoj mašini

- *Interproces*, ili *cross-proces*, komunikacija se odnosi na komunikaciju između dva odvojena procesa koja se nalaze na istom računaru.
- *Intraproces*, ili *in-proces*, komunikacija se odnosi na komunikaciju između dve softverske komponente koje rade unutar istog procesa.
- Zajedno ovi tipovi komunikacija čine ono što zovemo *local-machine* komunikacija. WCF ne pravi razliku izbeđu *intraprocesa i interprocesa*, umesto razlika, WCF nudi jedinstven *on-machine* transportni kanal zasnovan na named pipes**.**

- *Named pipes* su standardno sredstvo za međuprocesnu komunikaciju (IPC) na Windows, kao i na Unix platformama. Preformanse *Named pipes* su zadovoljavajuće za većinu korisnika, ali ako ste nezadovoljni uvek postoji mogućnost izrade sopstvenog binding-a.

netNamedPipeBiding

- WCF podržava *interproces* i *intraporces* komunikaciju sa *netNamedPipeBinding-om* koji je baziran na *named pipes* transportu. Ovo je odličan binding za korišćenje pri interprocesnoj komunikaciji zato što pravi značajnu razliku u preformansama u odnosu na ostale binding standarde koje WCF podržva.
- Iako je moguće koristiti *named pipes* u komunikaciji preko mreže, WCF onemogućava ovakav slučaj. Ovo znači da se *netNamedPipeBinding* ili bilo koji drugi *binding* koji se zasniva na njemu može iskoristiti **kao osiguranje da se aplikacija neće naći na mreži**. A to se ostvaruje pomoću dva mehanizma. Prvi, *NetworkSecurity* Identifier-u je zabranjen pristup *named pipe*-u. Drugo, ime *named pipe-a* je slučajno generisano i smešteno u memoriju tako da samo klijenti na istoj mašaini imaju pristup. Sledeći deo koda pokazije format adresiranja za *netNamedPipeBinding*.

- ***net.pipe://localhost/{service location}***
- ***Default port: 808***

Osobine netNamedPipeBiding - 1

Attribute Name	Description	Default
`closeTimeout`	The maximum time to wait for the connection to be closed.	00:01:00
`hostNameComparisonMode`	Specifies the method for hostname comparison when parsing URIs.	StrongWildCard
`maxBufferPoolSize`	Maximum size of any buffer pools used by the transport.	524,888
`maxBufferSize`	Maximum number of bytes used to buffer incoming messages in memory.	65,536
`maxConnections`	The maximum number of outbound or inbound connections. Outbound and inbound connections are counted separately.	10
`maxReceivedMessageSize`	The maximum size of an incoming message.	65,536

Osobine netNamedPipeBiding - 2

Attribute Name	Description	Default
name	The name of the binding.	
openTimeout	The maximum time to wait for an open connection operation to complete.	00:01:00
readerQuotas	Specify the complexity of messages that can be processed (for example, size).	n/a
receiveTimeout	The maximum time to wait for a receive operation to complete.	00:01:00
security	Specifies the security settings of the binding.	n/a
sendTimeout	The maximum time to wait for a send operation to complete.	00:01:00
transactionFlow	Enable transactions to flow from the client to the server.	false
transactionProtocol	The type of transactions supported either OleTransactions or WSAtomicTransactions.	OleTransactions

Konfigurisanje servera za netNamedPipeBiding

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service name="EssentialWCF.StockQuoteService">
        <host>
          <baseAddresses>
            <add baseAddress="net.pipe://localhost/stockquoteservice" />
          </baseAddresses>
        </host>
        <endpoint address=""
                  contract="EssentialWCF.IStockQuoteService"
                  binding="netNamedPipeBinding" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Konfigurisanje klijenta za netNamedPipeBiding

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <client>
      <endpoint address="net.pipe://localhost/stockquoteservice"
                binding="netNamedPipeBinding"
                contract="EssentialWCF.IStockQuoteService">
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

Interoperabilno povezivanje

- Web servisi su osnova za interoperabilnu komunikaciju između heterogenih sistema. Na primer, servisi napravljeni na Java platformi, kao na primer IBM Webshere ili BEA WebLogic moraju komunicirati sa klijentima napravljenim na .NET platformi. Isto tako, servisi napravljeni na .NET platformi moraju komunicirati sa Java klijentima. Pre WCF-a, ASP.NET Web Services (ASMX) i Web Services Enhancements (WSE) su podržavali ovakvu mogućnost. Sa dolaskom .NET 3.0 uveden je WCF koji je direktna zamena za ove tehnologije. Dodatan plus je to što se sve to nalazi u jednom *framework*-u za pravljenje Web servisa. WCF sadrži nekoliko binding-a pomoću kojih je otkrivanje interoperabilnih Web servisa moguće, a to su:
  - *basicHttpBinding,*
  - *wsHttpBinding,*
  - *wsDualHttpBinding, i*
  - *wsFederationHttpBinding.*

basicHttpBinding

- **basicHttpBinding** podržava WS komunikaciju baziranu na WS-I Basic Profile 1.1 specifikaciji. Ovo ukuljučuje standarde kao što su SOAP 1.1, WSDL 1.1 i Mesagge Security 1.0 (uključujući X.509 i UserName Token Profile v 1.0). Iako basicHttpBinding omogućava interoperabilnost kroz heterogene sisteme, on nema podršku za najnovije Web service standarde kao što su *transactions* i *reliable messaging*. *basicHttpBinding* je predviđen za korišćenje na aplikacijama koje koriste Web servise bazirane na WS-BP 1.1 specifikaciji kao što je ASP.NET ASMX Web Service.
- To znači da je basicHttpBinding configurisan da radi sa starijim standardima kao što je SOAP 1.1. On je i jedini binding koji po početnoj konfiguraciji nije siguran. Još jedan nedostatak ovog bindinga je dostava poruka po određenom redosledu. Ovo znači da kada klijent pošalje više poruka ka servisu, nije garantovano da će one stići istim redosledom kojim su poslate. Sledeći deo koda pokazuje format adresiranja za basicHttpBinding.
- `http://{hostname}[:port]/{service location}`
- `https://{hostname}[:port]/{service location}`
- Podrazumevani port: `80` za `http` i `443` za `https`

Osobine *basicHttpBinding* - 1

Attribute Name	Description	Default
`bypassProxyOnLocal`	Bypass the proxy settings for local endpoints.	false
`closeTimeout`	The maximum time to wait for the connection to be closed.	00:01:00
`hostNameComparisonMode`	Specifies the method for hostname comparison when parsing URIs.	StrongWildCard
`maxBufferPoolSize`	Maximum size of any buffer pools used by the transport.	524,888
`maxBufferSize`	Maximum number of bytes used to buffer incoming messages in memory.	65,536
`maxReceivedMessageSize`	The maximum size of an incoming message.	65,536
`messageEncoding`	The type of encoding used to encode messages.	Text
`name`	The name of the binding.	

Osobine *basicHttpBinding* - 2

Attribute Name	Description	Default
`name`	The name of the binding.	
`openTimeout`	The maximum time to wait for an open connection operation to complete.	00:01:00
`proxyAddress`	Specify a specific Web proxy to use. `useDefaultWebProxy` must be false for this setting to apply.	n/a
`readerQuotas`	Specify the complexity of messages that can be processed (for example, size).	n/a
`receiveTimeout`	The maximum time to wait for a receive operation to complete.	00:01:00
`security`	Specifies the security settings of the binding.	n/a
`sendTimeout`	The maximum time to wait for a send operation to complete.	00:01:00
`textEncoding`	The method of character encoding used to encode messages. `messageEncoding` must be set to Text for this setting to apply.	utf-8
`transferMode`	Determines how messages are sent across the network. Messages can either be buffered or streamed.	Buffered
`useDefaultWebProxy`	Use the default Web proxy specified by the operating system.	true

Konfigurisanje servera za *basicHttpBinding*

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service name="EssentialWCF.StockQuoteService">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost/stockquoteservice" />
          </baseAddresses>
        </host>
        <endpoint address=""
                  contract="EssentialWCF.IStockQuoteService"
                  binding="basicHttpBinding" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Konfigurisanje klijenta za *basicHttpBinding*

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <client>
      <endpoint address="http://localhost/stockquoteservice"
                binding="basicHttpBinding"
                contract="EssentialWCF.IStockQuoteService">
      </endpoint>
    </client>
  </system.serviceModel>

</configuration>
```

Napredni Web Servisi - wsHttpBinding

- *N*apredni Web servisi (engl. *Advanced Web services*) izlažu specifikacije sa oznakom **WS-***, (*wsstar).*
- WCF ima podršku za WS-* specifikacije uključujući sigurnost, pouzdanost poruka i transakcije.
- Podrška za WS-* je uključena kroz WCF *framework*, a *wsHttpBinding* je primer te podrške. Ovaj *binding* omogućava interoperabilnu komunikaciju kroz heterogene sisteme, a i obezbeđuje sigurnost, transakcije, i pouzdano slanje/primanje poruka. wsHttpBinding je default binding u .NET 3.0. Format adresiranja za wsHttpBinding:

- `http://{hostname}[:port]/{service location}`
- `https://{hostname}[:port]/{service location}`
- Podrazumevani port ***: 80 (http)***
- Podrazumevani port ***: 443 (https)***

Osobine *wsHttpBinding* - 1

Standard	Description
SOAP 1.2	Lightweight protocol for exchange of information in a decentralized, distributed environment
WS-Addressing 2005/08	Transport-neutral mechanisms to address Web services and messages
WSS Message Security 1.0	Specification for securing Web services using a variety of mechanisms such as PKI, Kerberos, and SSL
WSS Message Security UsernameToken Profile 1.1	Support for security tokens based on a username and optionally a password (or password equivalent such as a shared secret)
WSS SOAP Message Security X509 Token Profile 1.1	Support for tokens based on X.509 certificates
WS-SecureConversation	Extensions to WS-Security to provide a secure context for multiple message exchanges

Osobine *wsHttpBinding* - 2

Standard	Description
WS-Trust	Extensions to WS-Security to request and issue tokens and to manage trust relationships
WS-SecurityPolicy	Policy assertions for WS-Security, WS-Secure-Conversation, and WS-Trust, which are expressed using WS-Policy
WS-ReliableMessaging	A protocol for guaranteeing messages are deliv-ered, properly ordered, and received without duplication
WS-Coordination	A framework for providing protocols that coordi-nate the actions of distributed applications
WS-Atomic Transactions	A protocol that coordinates the actions of distributed applications based on the atomic transactions
WS-Addressing	A transport-neutral mechanism for addressing Web services

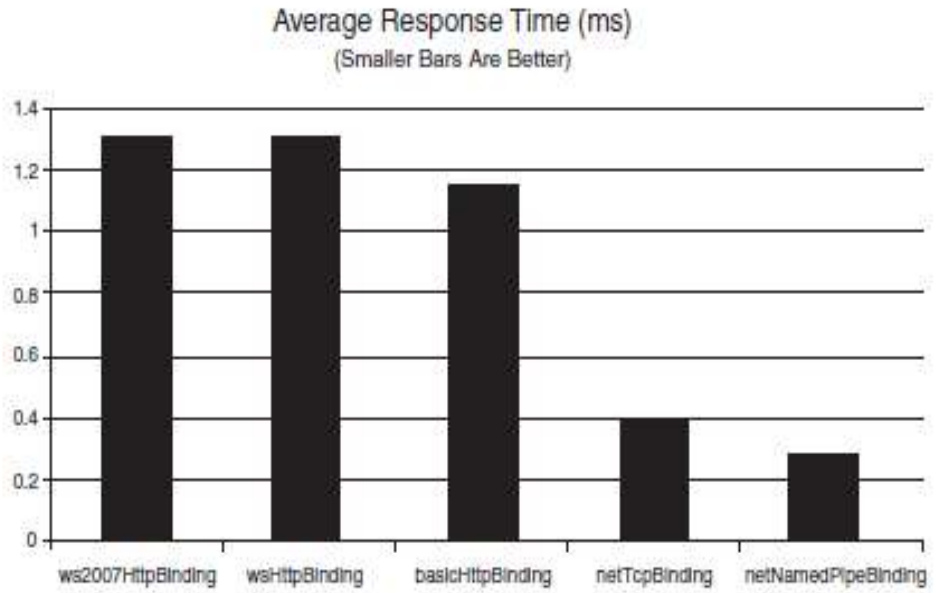
Poređenje različitih vrsta povezivanja

Uvod

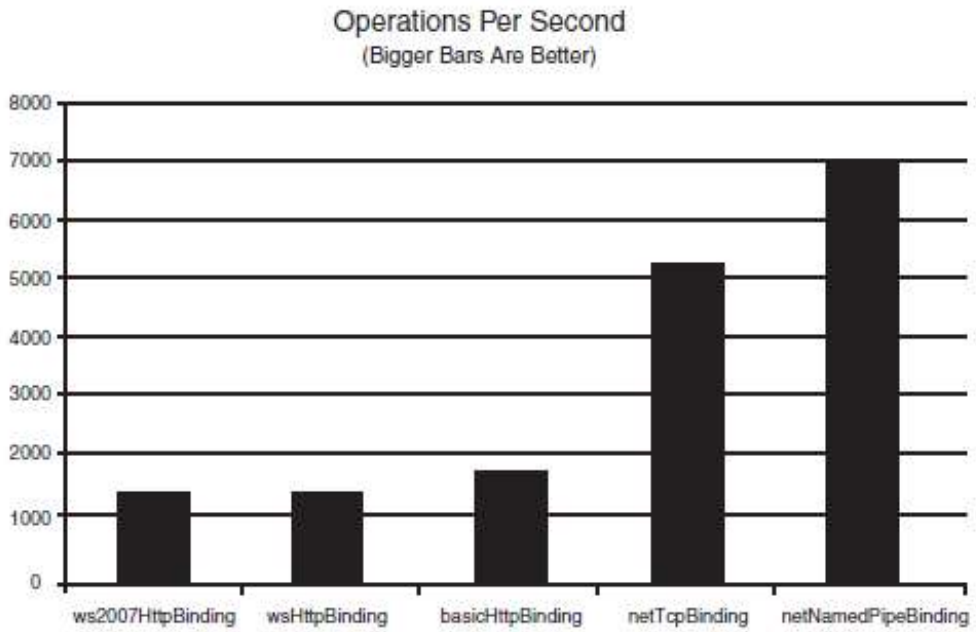
- Programeri moraju da znaju o preformansama i skalabilnosti *bindinga* kao i njihove karakteristike. One su jako važne za rad sa realnim aplikacijama gde su *service-level* ugovori i korisnički ugođaj jako bitni. Za tu potrebu testiran je učinak *bindinga* u WCF-u. Test operacija je veoma jednostavan, sastoji se od metode koja vraća 256 karaktera.
- Servis je otvoren kroz četiri različita bindinga: *netNamedPipeBinding*, *netTcpBinding*, *wsHttpBinding*, i *basicHttpBinding*. „Test klijent“ poziva *Get256Byts* operaciju 50.000 puta uzastopno kako bi potom uporedio rezultate različitih bindinga. Zatim je mereno prosečno vreme za koje se test završi u operaciji po sekundi, i mereno je CPU vreme. Svaki test se obavljao na istoj mašini koja je bila i servis i klijent. Ovo je rađeno kako bi lakše mogli da poredimo rezultate *bindinga*.

Prosečno vreme odgovora

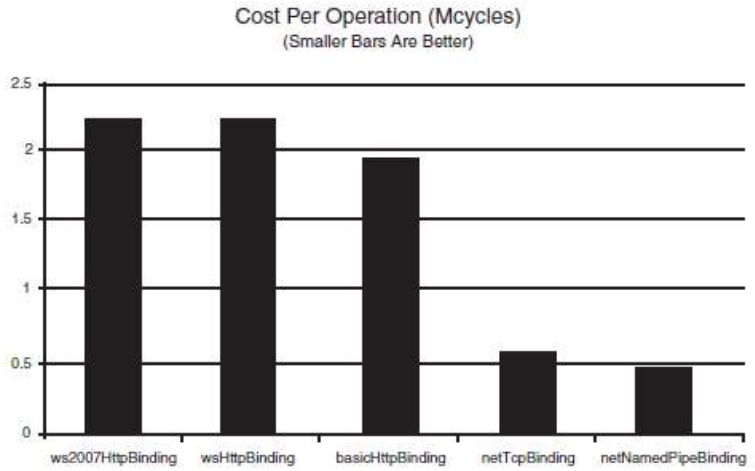
- Slika pokazuje prosečno vreme odgovora za svaku vrstu povezivanja.



- Naredna slika pokazuje prosečni broj operacija po sekundi za svaki binding. Ova merenja imaju uticaj na propusnost. Samo po jedna instanca klijenta je korišćena pri ovim testiranjima. Dodatna propusnost se može postići ako bi se koristili višestruki klijenti. “Operacije po sekundi” je mera koju smo koristili da odredimo skalabilnost.



- Za određivanje skalabilnosti bi trebalo isto tako uzeti u obzir i hardverske resurse koji se koriste za obradu svake operacije. Naredna slika pokazuje merenje skalabilnost tako što pokazuje cenu operacije u MCycles-ima. MCycles je mera bazirana na jačini CPU. Za potrebe ovog testa korišćen je Dell 4700 sa 3.4 GHz Pentium4 procesor, koji je jednak sa 3400 *MCycles* jedinica.



- Treba zapaziti da ws2007HttpBinding, wsHttpBinding, i basicHttpBinding imaju značajno veću cenu nego netTcpBinding ili netNamedPipeBinding

Ponašanja i odgovarajuća podešavanja servisa

Behaviors

Uvod

- Ponašanja *(Behaviors)* su **WCF klase koje utiču na operacije u vreme izvršavanja aplikacije** ili sistema.
- *Ponašanja* se aktiviraju pri startovanju izvršavanja WCF klijenta ili servisa i pri razmeni poruka.
- Služe za implementaciju opcija ugrađenih u WCF. Na primer, *ServiceHost* je odgovoran za instanciranje i konkurentnost servisa, uz to je odgovoran i za slanje poruka odgovarajućim operacijama.
- Postoje tri tipa behavior-a:
- ***Service Behavior***, koji radi na servisnom nivou i ima dozvoljen pristup svim endpoint-ovima. On kontroliše stvari kao što su instanciranje i transakcije. Service behavior takođe može da kontroliše autorizaciju i reviziju.
- ***Endpoint Behavior***, koji služi za instanciranje i odrađivanje akcija nad porukama, pri odlasku i dolasku na servis.
- ***Operation Behavior***, se pokreću na operativnom nivou i služi za manipulaciju serializaciom, transakcijama i u radu sa parametrima servisnih operacija.
- Postoji i ***Callback Behavior***, koji je sličan sa *service behavior*-om, sa tom razlikom što on kontroliše endpoint-ove na klijentskoj strani pri duplex komunikacijama.

Inicijalizacija

- Izvodi se pomoću klase ***ChanalFactory*** na klijentskoj strani, odnosno pomoću ***ServiceHost*** na servisnoj. Obe klase imaju slične funkcije.
  1. Prihvatanje .NET tipova kao ulaz i čitanje atributa.
  2. Učitavanje konfiguracije iz app.config ili web.config fajlova. Na klijentskoj strani *ChanalFactor*y očekuje binding informacije, dok na servisnoj strani *ServiceHost* očekuje kontakt i binding informacije.
  3. Izgradnja strukture pri izvršavanju, *ServiceDescription*.
  4. Početak komunikacije. Na strani klijenta, *ChanalFactory* koristi kanal za konekciju sa servisom. Na servisnoj strani, *ServiceHost* otvara kanal i sluša da li pristižu poruke.
- U koraku 1. behavior atribut je definisan u kodu, kao na primer: *[ServiceBehavior(TransactionTimeout="00:00:30")].*
- U koraku 2. behavior informacije se učitavaju iz konfiguracionog fajla.
- U toku koraka 3., *ChanalFactory* i *ServiceHost* klase stvaraju WCF izvršni deo i postaju zaduženi za ubacivanje ponašanja nađenih u prvom i drugom koraku u proces izvršavanja. Takođe, u koraku 3. ponašanje se može ručno dodati u kod, kao na primer:

  Endpoint.Behaviors.Add(new MyBehavior());

- Pored inicijalizacije, ***ponašanje*** **može da operiše nad podacima pre prenosa ili posle prijema**.

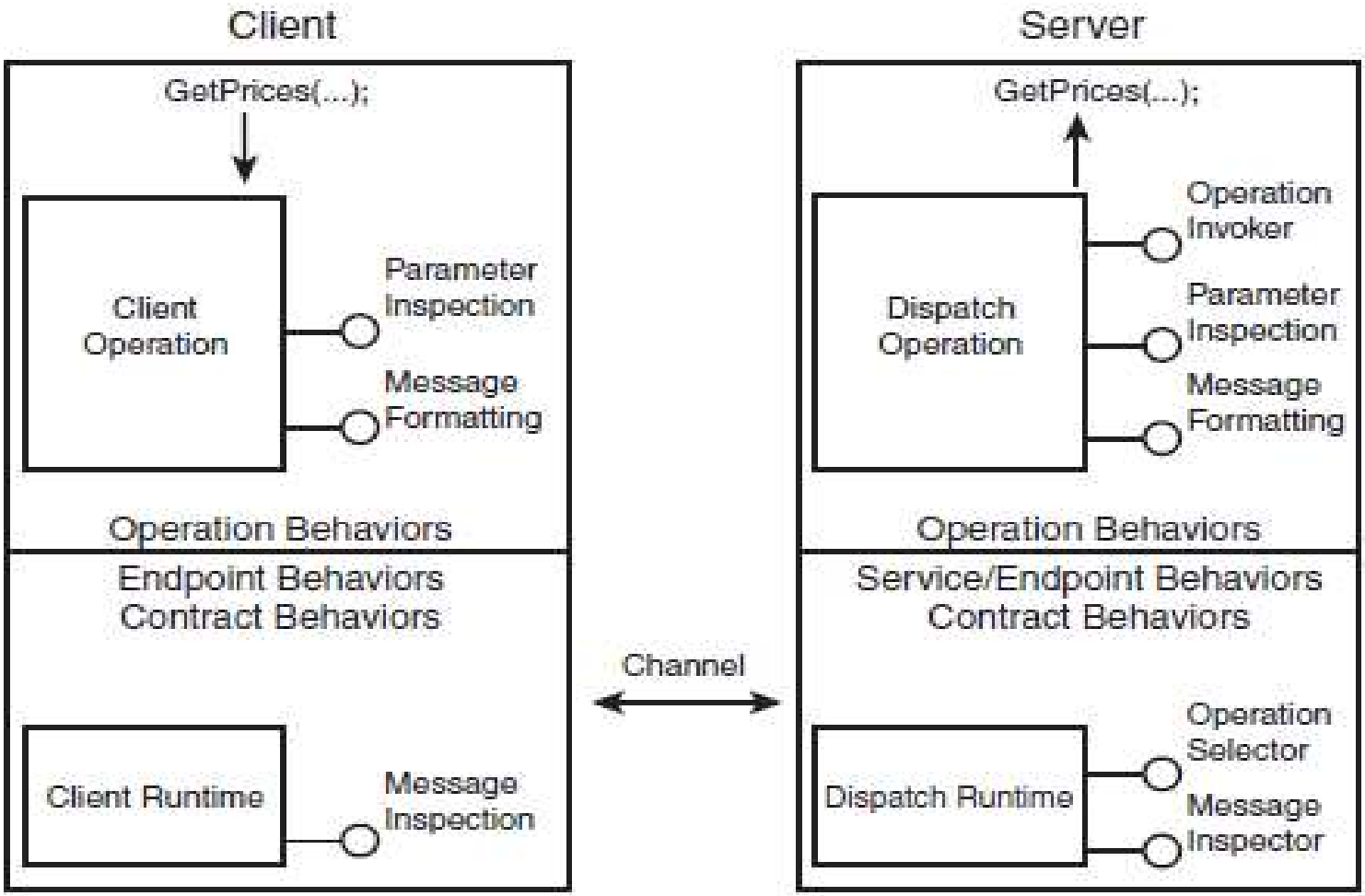
❑ Na klijentskoj strani, *behavior* se može koristiti za tri stvari:

- **Parameter Inspection** - Pregleda ili menja podatke u njihove .NET reprezentacije, pre konvertovanja u XML.
- **Message Formatting** - Pregleda ili menja podatke tokom konverzije iz .NET tipe u XML.
- **Message Inspection** - Pregleda ili menja podatke u njihovoj XML reprezentaciji, pre nego što budu pretvoreni u .NET tipove.

❑ Na servisnoj strani, behavior se može koristiti za dve stvari:

- **Operation Selection** - Na servisnom nivou, on pregleda pristigle poruke i određuje koju operaciju treba pozvati.
- **Operation Invocation** - Na operacionom nivou - pokreće metodu određene klase.

Behavior elementi



- Slika opisuje protok kontole među behavior elemenata koji su aktivirani tokom razmene poruka između klijenta i servisa.

- Kada aplikaconi kod na klijentskoj strani pozove GetPrice(), *ParameterInspector* i *MessageFormatter* su pozvani i prosleđuju im se parametri u njihovom .NET formatu. Zatim, takodje na klijentskoj strani, *MessageInspector* se poziva i prosleđuje mu se XML poruka. Na strani servisa, kada poruka stigne u kanal, *MessageInspector* i *OperationSelector* su pozvani i prosleđuje im se pristigla poruka kako bi je pregledali i odredili koja operacije treba da je primi. Zatim se poziva *MessageFormatter* kako bi formatirao poruku kao .NET tip, a potom se poziva i *ParameterInspector* i njemu se prosleđuje poruka u .NET reprezentaciji. Najzad, *OperationInvoker* je pozvan da pokrene metodu u odgovarajućoj klasi.

Konkurentnost i instanciranje

- **Konkurentnost** je mera koliko zadataka se može odraditi u isto vreme, a merna jedinica su operacije (zadaci, poslovi, transakcije i slično).

- **Vreme izvršavanja** (*Execution time*) je mera koliko dugo je potrebno da se neki posao završi i meri se u milisekundama (sekundama, minutima, satima, danima).

- **Propusnost** (*Throughput*) je mera koliko zadataka se može izvršiti u zadatom vremenskom intervalu. Postoje dva načina za povećanje propusnosti: smanjiti vreme izvršavanja ili povećati konkurentnost. Smanjenje vremena izvršavanja se može izvršiti tako što bi se promenio algoritam koji radi na zadatku ili dodavanjem dodatnih hardverskih resursa. To znači da WCF ne može mnogo da pomogne u ovom slučaju. Konkurentnost se može povećati tako što bi se zadaci izvršavali paralelno. WCF poseduje dva behavior-a za ovu svrhu: ***InstanceContextMode*** i ***ConcurrencyMode*****.**

- ***InstanceContexMode*** behavior se koristi za kontrolu instanciranja i može se podesiti na jednu od tri vrednosti:

✓ **Single** - Jedna instanca servisne klase odrađuje sve dolazeće zahteve. Ova implementacija se naziva i Singlton.

✓ **PerCall** - Jedna instanca servisne klase je kreirana po svakom zahtevu.

✓ **PerSession** - Jedna instanca servisne klase je kreirana po klijentskoj sesiji.

Podrazumevano podešavanje je ***PerSession***, i nalaže WCFu da kreira novu instancu servisne klase za svaki proxy.

- ***ConcurrencyMode** behavor* se koristi za kontrolu konkurentnosti niti sa instancom servisa. Početno podešavanje je tipa ***Single*** i nalaže WCF-u da odradi niti redom jednu po jednu po servisnoj instanci.
- Moguće su tri vrednosti, koje se mogu podesiti:

✓ **Single** - Samo jedna po jedna nit se može odraditi. Ovo je najsigurnije podešavanje zato što servisna operacija ne mora da brine oko bezbednosti niti.

✓ **Reentrant** - Samo jedna po jedna nit može odraditi servisnu klasu, ali nit može napustiti obradu i vratiti se kasnije da je dovrši.

✓ **Multiple** - Višestruki broj niti može da ima pristup servisu u isto vreme. Ova opcija zahteva da servisna klasa bude napravljena kao bezbedna za niti.

- Korišćenjem *InstanceContextMode* i *ConcurrencyMode*, zajedno omogućava izradu posebnih servisa koji mogu da zadovolje najrazličitije potrebe

Primer 1. Podrazumevano ponašanje

Servis.

```
[ServiceContract]
interface ICenovnik
{
    [OperationContract]
    string cenaZaSifru(int sifra);
}

class Cenovnik : ICenovnik
{
    Cenovnik()
    {
        Console.WriteLine("Servis pokrenut:"+DateTime.Now);
    }

    public string cenaZaSifru(int sifra)
    {
        Console.WriteLine("" + DateTime.Now + ": cenaZaSifru pozvano u niti " +
Thread.CurrentThread.ManagedThreadId);
        Thread.Sleep(5000);
        return "cena je " + 555.33;
    }
}
```

- Konfiguracioni fajl servisa:

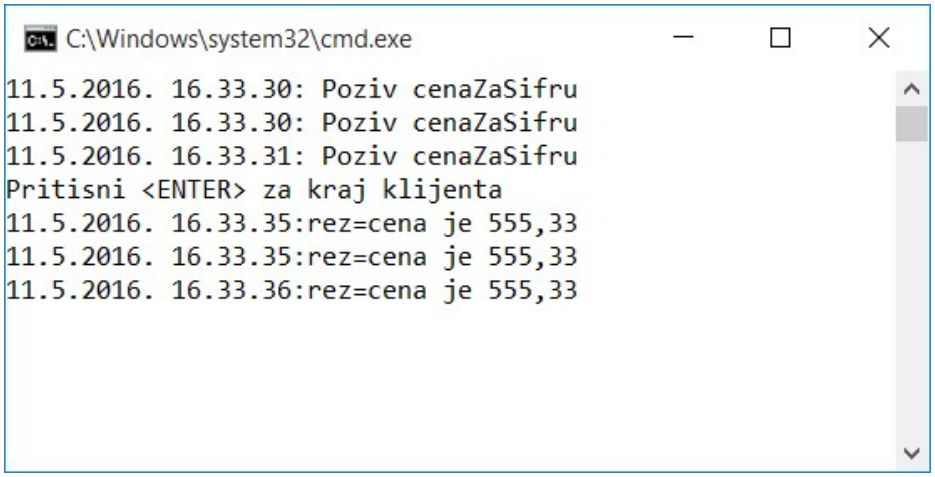
```
<system.serviceModel>
  <services>
    <service name="ist10Primer1.Cenovnik" behaviorConfiguration="myServiceBehavior">
      <host>
        <baseAddresses>
          <add baseAddress="http://localhost:8733/cenovnik"/>
        </baseAddresses>
      </host>
      <endpoint address="" binding="basicHttpBinding" contract="ist10Primer1.ICenovnik" />
      <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />

    </service>
  </services>

  <behaviors>
    <serviceBehaviors>
      <behavior name="myServiceBehavior">
        <serviceMetadata httpGetEnabled="true"/>
      </behavior>
    </serviceBehaviors>
  </behaviors>

</system.serviceModel>
```

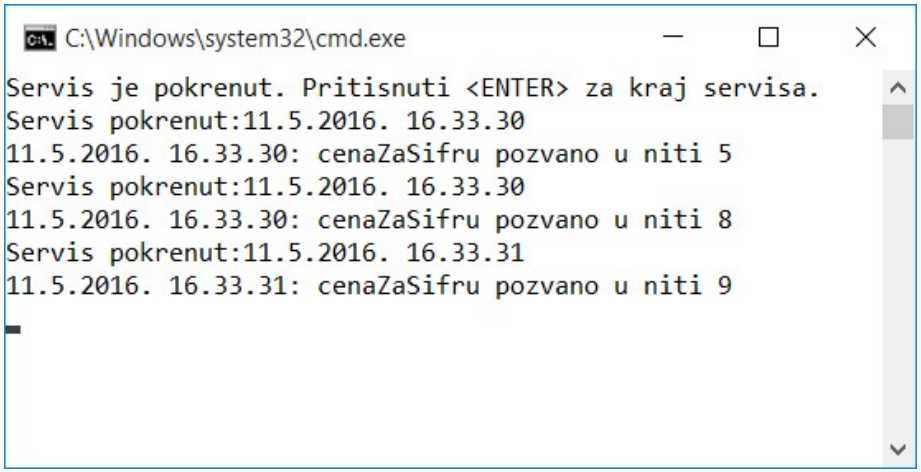
Klijent

```
class Klijent
{
    static void Main(string[] args)
    {
        radSaCenovnikom();
        Thread.Sleep(100);
        radSaCenovnikom();
        Thread.Sleep(100);
        radSaCenovnikom();
        Thread.Sleep(100);

        Console.WriteLine("Pritisni <ENTER> za kraj klijenta");
        Console.ReadLine();
    }

    async public static void radSaCenovnikom()
    {
        ServiceReference1.CenovnikClient proxy = new
ServiceReference1.CenovnikClient();

        Console.WriteLine("" + DateTime.Now + ": Poziv cenaZaSifru");
        string rez = await proxy.cenaZaSifruAsync(123);
        Console.WriteLine("" + DateTime.Now + ":" + "rez=" + rez);
    }
}
```


Napomena: Možete probati i sa prosleđivanjem istog proxy objekta metodi radSaCenovnikom ;

- Izlaz: Client (levo), Server (desno)





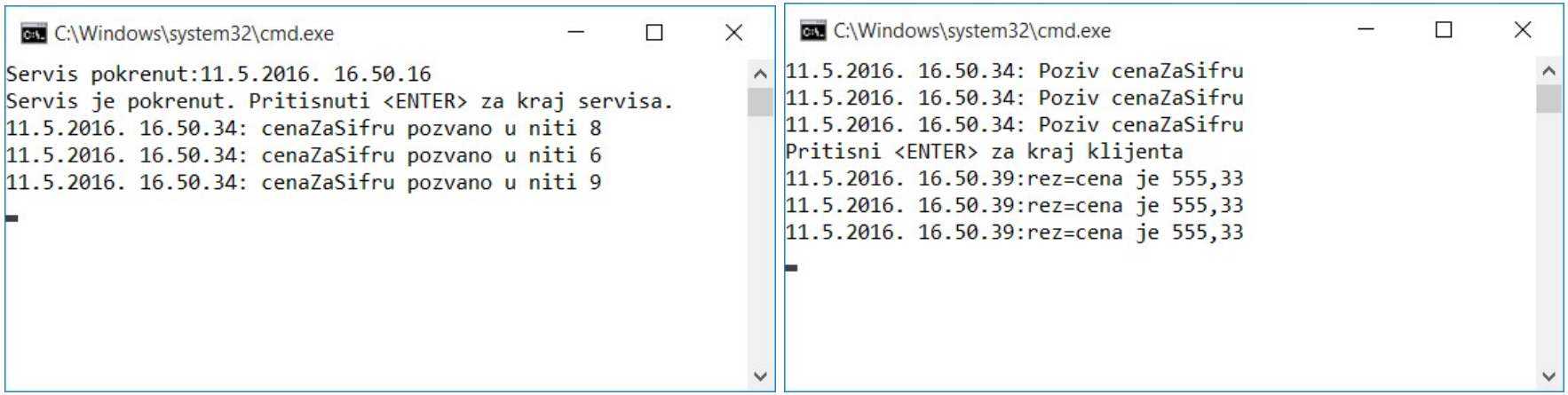
Svaka instanca servisa se kreira na zahtev klijenta i svaki zahtev se obrađuje u zasebnoj niti. Pošto basicHttpBinding ne podržava sesije, umesto *PerSession* koristi se *PerCall*

Primer 2. Više niti, jedna instanca

- Jedna nit po instanci je podrazumevano ponašanje. Ipak, ako je konstruktor jako skupa operacija (na pr. učitavanje podataka iz baze) svakako da nije dobro kreiranje nove instance za svaki poziv. Da bi kreirali jednu instancu servisa koju bi delile niti **treba koristiti *InstanceContextMode.Single* zajedno sa *ConcurrencyMode.Multiple***. Servisni kod mora obrađivati sinhronizaciju kako bi obezbedio skladištenje lokalnih niti.

```
[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single,
ConcurrencyMode = ConcurrencyMode.Multiple)]
class Cenovnik : Icenovnik
. . . . . .
```

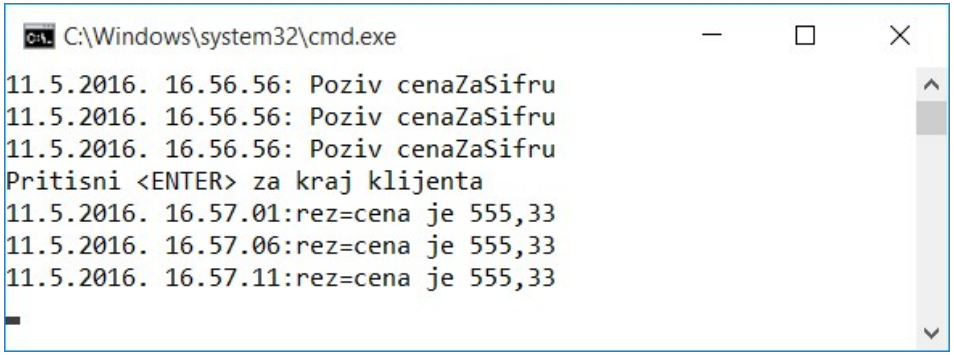
- Obratite pažnju gde stoje atributi

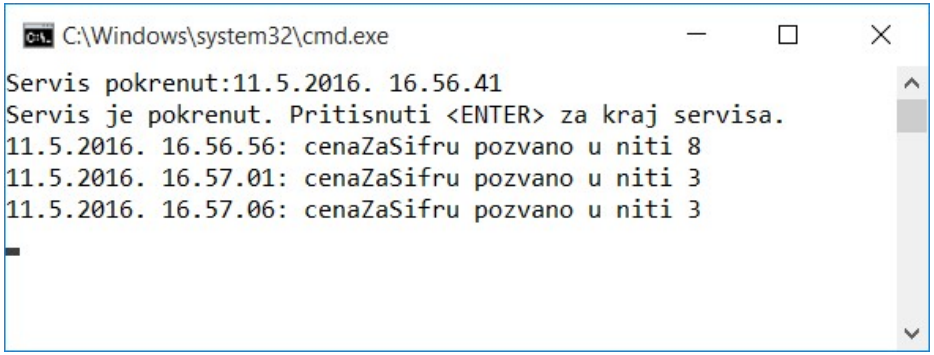


- Samo je jedna instanca servisa kreirana, a pokrenuto je 3 niti.

Primer 3. Singleton (1 instanca 1 nit)

```
[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single, ConcurrencyMode =
ConcurrencyMode.Single)]
class Cenovnik : ICenovnik
```





Kreira se 1 nit u jednom trenutku.

Napomena: Ne garantuje se da će biti kreirana samo 1 ista nit.

Upotreba sesija

- Uloga sesija je da:
- Čuvaju podatke specifične za korisnika

- Da bi je koristili, potrebno je:

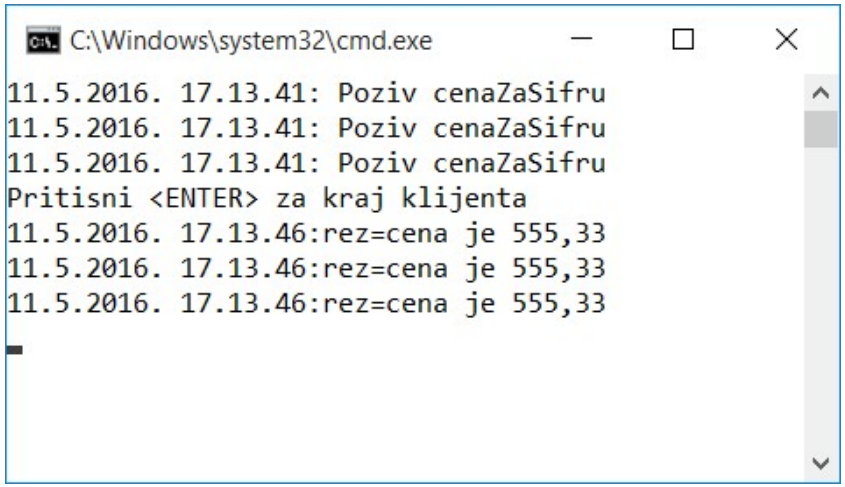
1. Zahtevati sesiju u ugovoru
2. Omogućiti sesiju u servisu

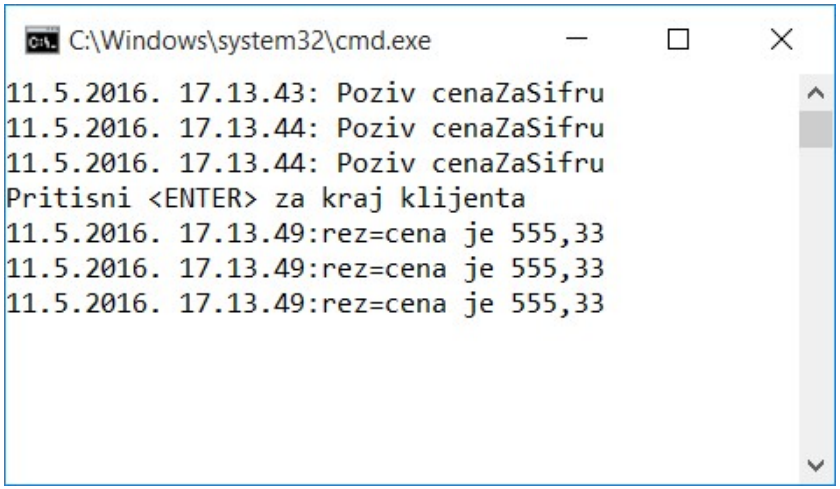
Voditi računa o vrsti povezivanja

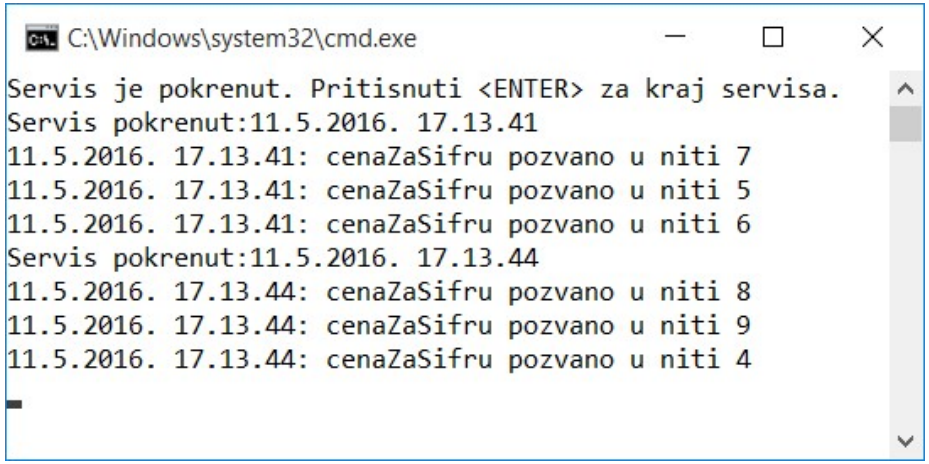
[ServiceContract(SessionMode = SessionMode.Required)]

[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession, ConcurrencyMode = ConcurrencyMode.Multiple)]

```
[DataContract]
class StockPrice
{
        [DataMember]
        public double price;
        [DataMember]
        public int calls;
}

[ServiceContract(SessionMode = SessionMode.Required)]
interface IStockService
{
        [OperationContract]
        StockPrice GetPrice(string ticker);
}

[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession,
                 ConcurrencyMode = ConcurrencyMode.Multiple)]
class StockService : IStockService
{
        System.Object lockThis = new System.Object();
        private int n_Calls = 0;
        StockService()
        {
            Console.WriteLine("{0}: Created new instance of
                                    StockService on thread",
                System.DateTime.Now);
        }
        public StockPrice GetPrice(string ticker)
        {
            StockPrice p = new StockPrice();
            Console.WriteLine("{0}: GetPrice called on thread {1}",
              System.DateTime.Now,
              Thread.CurrentThread.ManagedThreadId);
            p.price = 94.85;
            lock (lockThis)
            {
                p.calls = ++n_Calls;
            }
            Thread.Sleep(5000);
            return (p);
        }
}
```







- *Kreirane su 2 instance za svakog klijenta po jedna. Podaci se čuvaju u sesiji pa su i brojaci nezavisni za svakog klijeta.*

Kontrolisanje broja konkurentih instanci

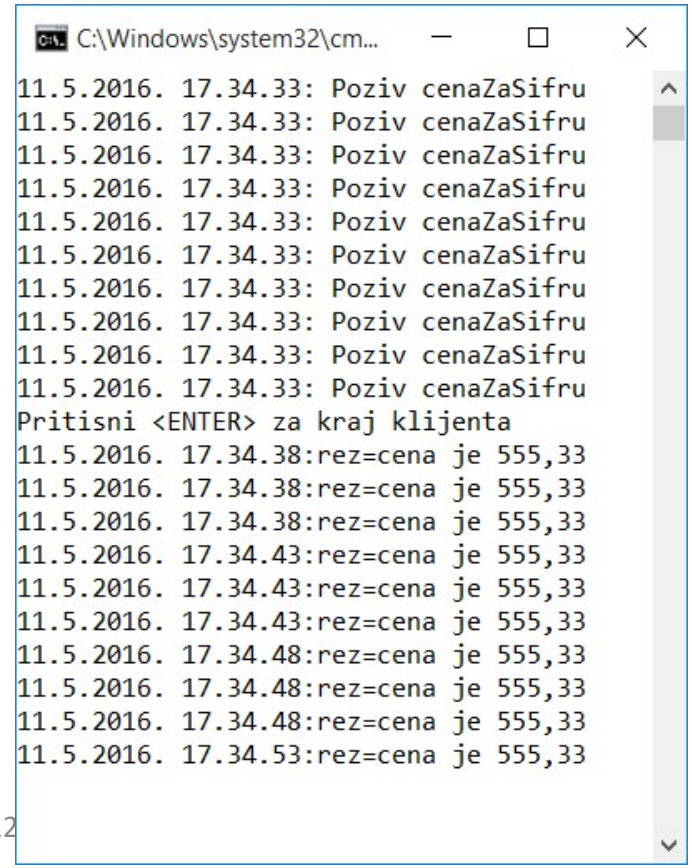
- Podrazumevano, WCF podiže onoliko instaci koliko je potrebno. Ako nije drugačije navedeno, WCF će kreirati serversku instacu za svaki dolazeći zahtev.

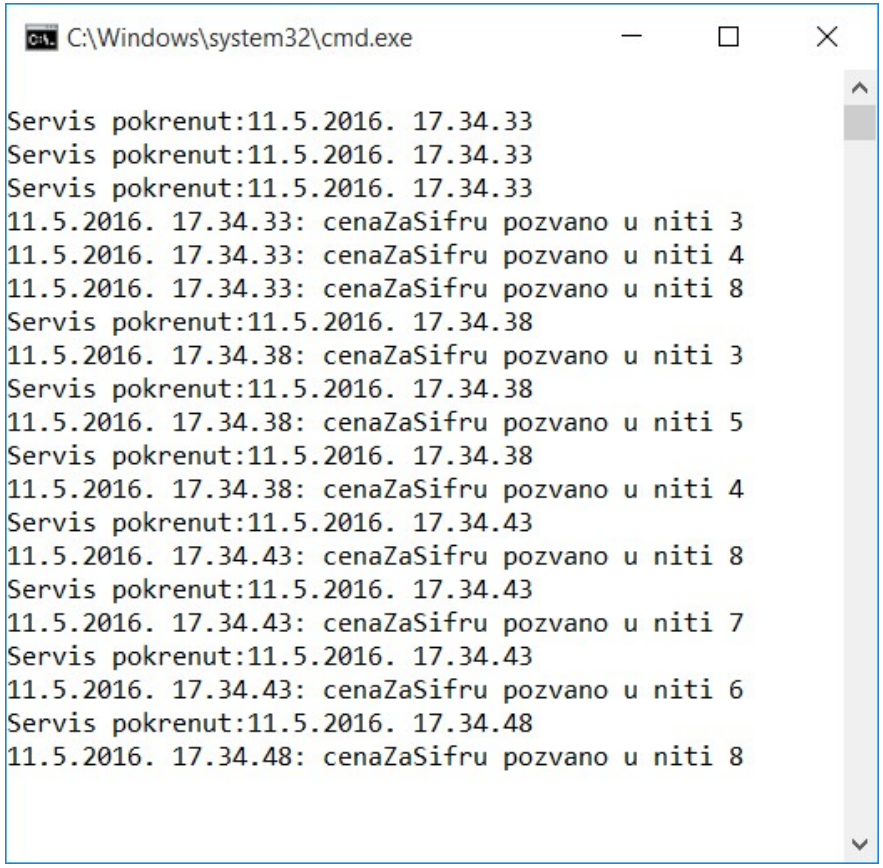
- Prevazilaženje mogućeg problema, usled nekontrolisanog povećanja broja instanci, se postiže kontrolom instaciranja preko elementa ***serviceThrottling.***

```
<system.serviceModel>
  <services>
    <service name="ist10Primer1.Cenovnik" behaviorConfiguration="myServiceBehavior">
      <host>
        <baseAddresses>
          <add baseAddress="http://localhost:8733/cenovnik"/>
        </baseAddresses>
      </host>
      <endpoint address="" binding="basicHttpBinding" contract="ist10Primer1.ICenovnik" />
      <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />

    </service>
  </services>

  <behaviors>
    <serviceBehaviors>
      <behavior name="myServiceBehavior">
        <serviceMetadata httpGetEnabled="true"/>
        <serviceThrottling maxConcurrentInstances="3" />
      </behavior>
    </serviceBehaviors>
  </behaviors>

</system.serviceModel>
```

Primer

- Kreriano je 10 klijenata. Servis traje 5 sek za svakog.
- Rezultat: 3 klijenata će biti obrađeno u prvih 5 sek, narednih 3 u drugih 5 sek itd.





Kontrolisanje broja konkurentnih poziva

```
[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single,
                 ConcurrencyMode = ConcurrencyMode.Multiple)]
public class StockService : IStockService
{
        StockService()
        {
```

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <system.serviceModel>

    <services>
      <service name="EssentialWCF.StockService"
           behaviorConfiguration="throttling">
    <host>
      <baseAddresses>
        <add baseAddress="http://localhost:8000/EssentialWCF"/>
      </baseAddresses>
    </host>
    <endpoint address=""
              binding="basicHttpBinding"
              contract="EssentialWCF.StockService" />
  </service>
</services>

<behaviors>
  <serviceBehaviors>
    <behavior name="throttling">
      <serviceThrottling maxConcurrentCalls="5"/>
    </behavior>
  </serviceBehaviors>
</behaviors>

  </system.serviceModel>
</configuration>
```

Kontrolisanje broja konkurentnih sesija

```
[ServiceContract(SessionMode = SessionMode.Required)]
public interface IStockService
{
        [OperationContract]
        double GetPrice(string ticker);
}

[ServiceBehavior(InstanceContextMode = InstanceContextMode.P
                 ConcurrencyMode = ConcurrencyMode.Multiple)
public class StockService : IStockService
{
        StockService()
```

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <system.serviceModel>

    <services>
      <service name="EssentialWCF.StockService"
               behaviorConfiguration="throttling">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8000/EssentialWCF"/>
          </baseAddresses>
        </host>
        <endpoint address=""
                  binding="wsHttpBinding"
                  contract="EssentialWCF.StockService" />
      </service>
    </services>

    <behaviors>
      <serviceBehaviors>
        <behavior name="throttling">
          <serviceThrottling maxConcurrentSessions="5"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>

  </system.serviceModel>
</configuration>
```