Message Passing Systems

Jim Fawcett

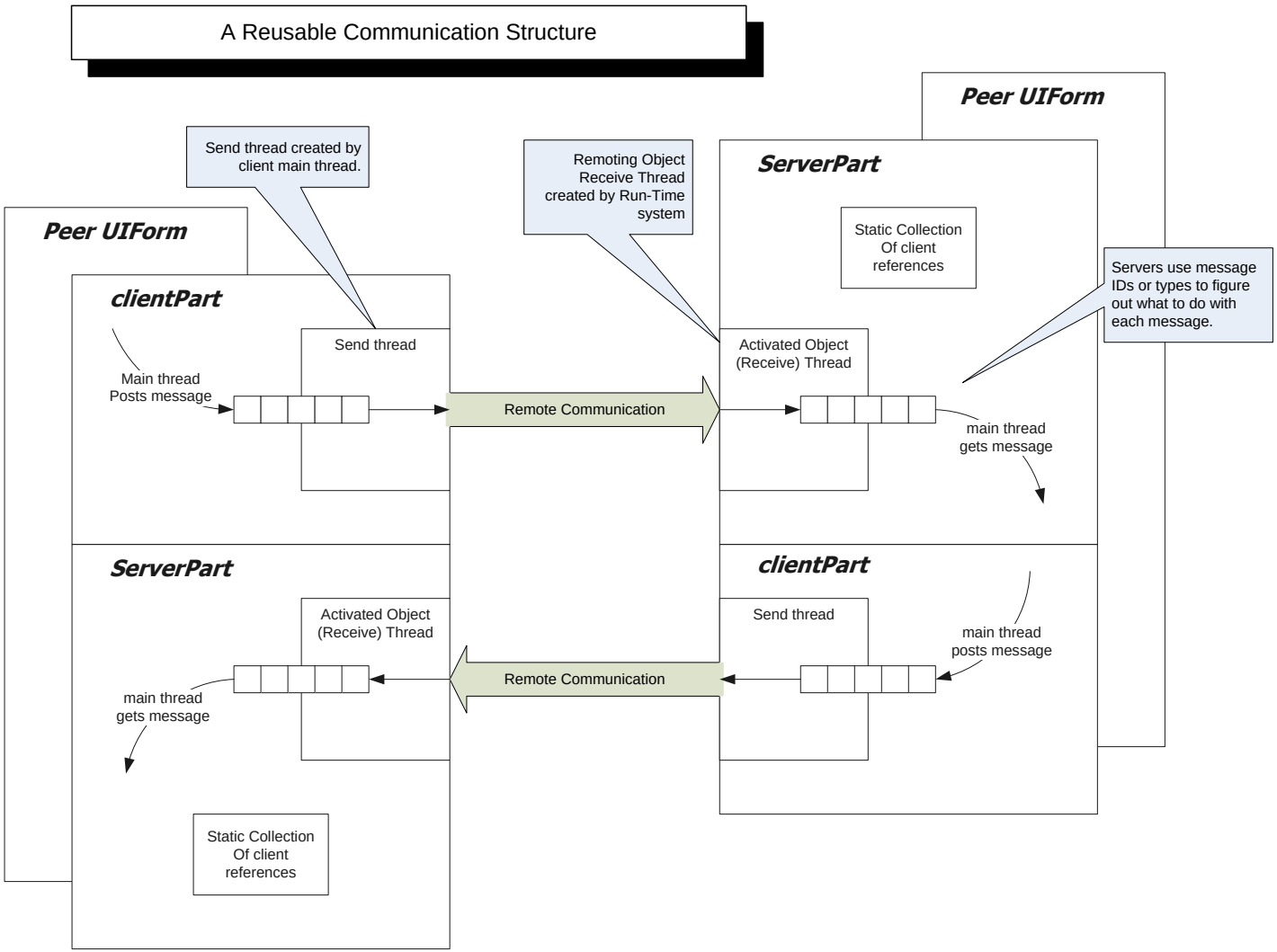
CSE681 – Software Modeling and Analysis

Summer 2007

Distributed Architectures

- Project #5 statement asks you to develop a complex collaboration system that requires communication between multiple servers and clients:
  - ASCS provides agents and servers for collaboration, managing source code and other project resources, and running extensive tests.
  - The ASCS requires a communication subsystem that can use either a corporate network or the internet as a transmission medium.
  - It could be designed in a Peer-To-Peer structure (see next slide) since all of the agents and servers act as both servers and clients. For example, the Repository Server acts as a client during check-in and a server during check-out. The same is true of the Collaboration Server, Testbed Server and each of the Clients and agents.
  - In Project #5, you need to investigate building a reusable communication layer using Message-Passing (MP) communication styles, which could be implemented with sockets, remoting, or web services.
    - Remoting and Web Services directly support an RPC style of communication using proxies and remote objects. But most designs using RPC are not reusable. The communication channel is tied to application specific remote objects. It can, with some work, be made reusable, as shown in the next few slides.
    - Message-passing can be layered on top of .Net Remoting or web services, as we will show later in this presentation. This can lead to a very general reusable communication layer.

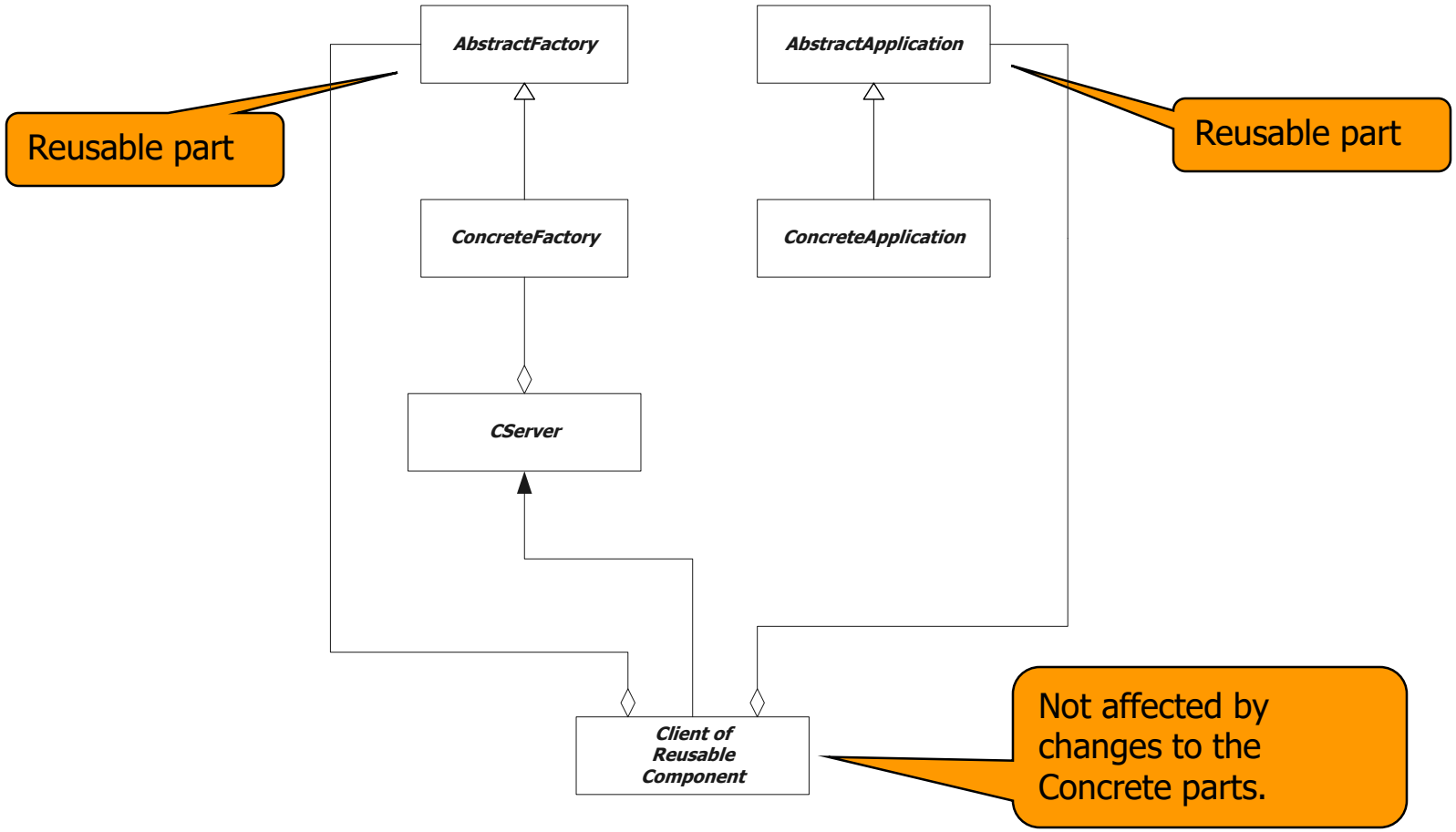
Peer-To-Peer Architecture



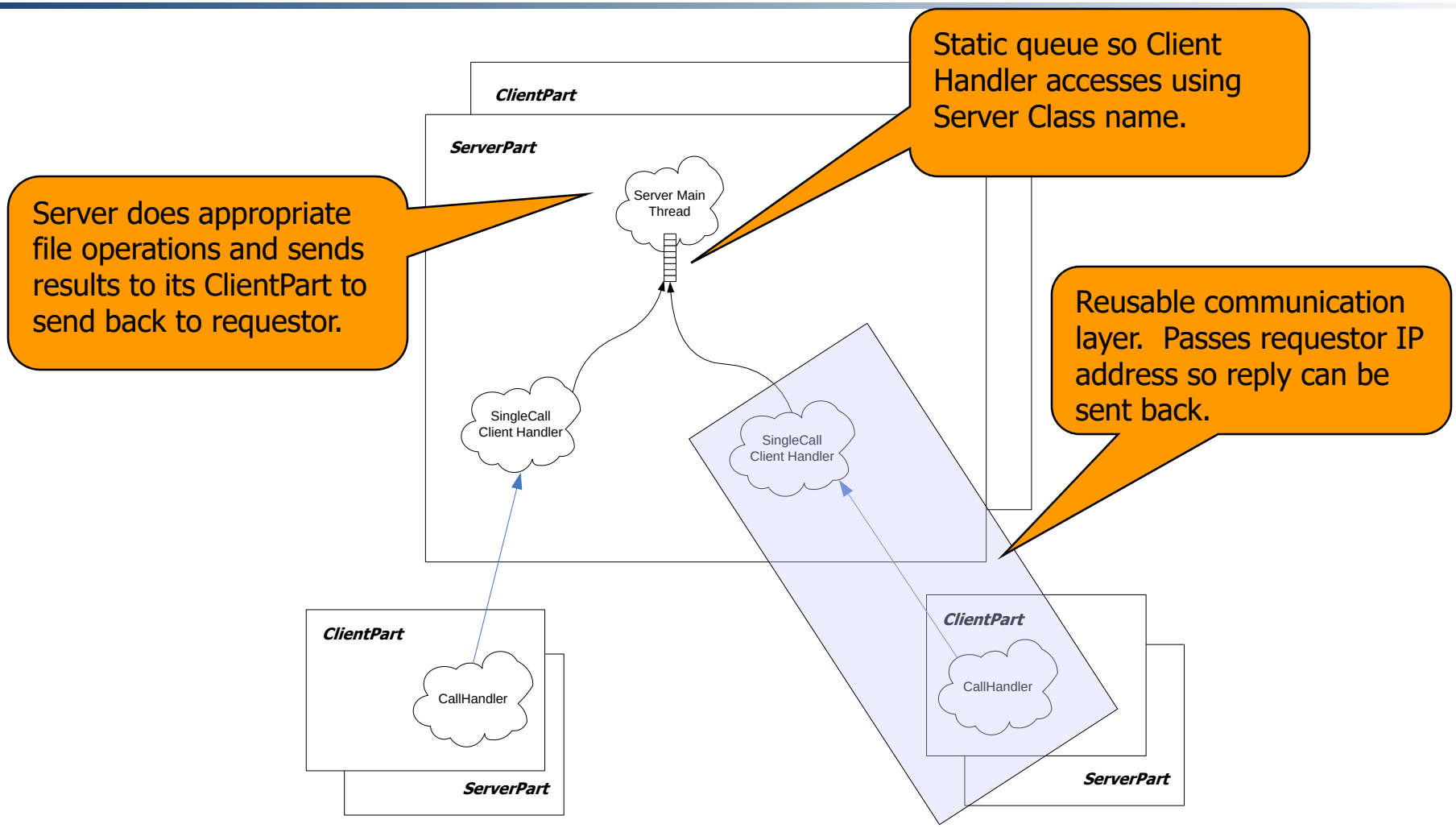
Reusable Communication Channel

- In order to build a reusable communication channel we have to isolate the channel from application specific code.
- We can do that using what is known as the Abstract Factory Pattern (discussed in CSE776 – Design Patterns, each summer).
  - In this pattern clients of application code can be isolated from the concrete classes that implement the application by using:
    - AbstractApplication class – defines the application interface
    - ConcreteApplication class – supplies application functionality
    - AbstractFactory class – allows clients to create objects bound to the abstract application interface.
    - ConcreteFactory class – creates instances of application specific classes

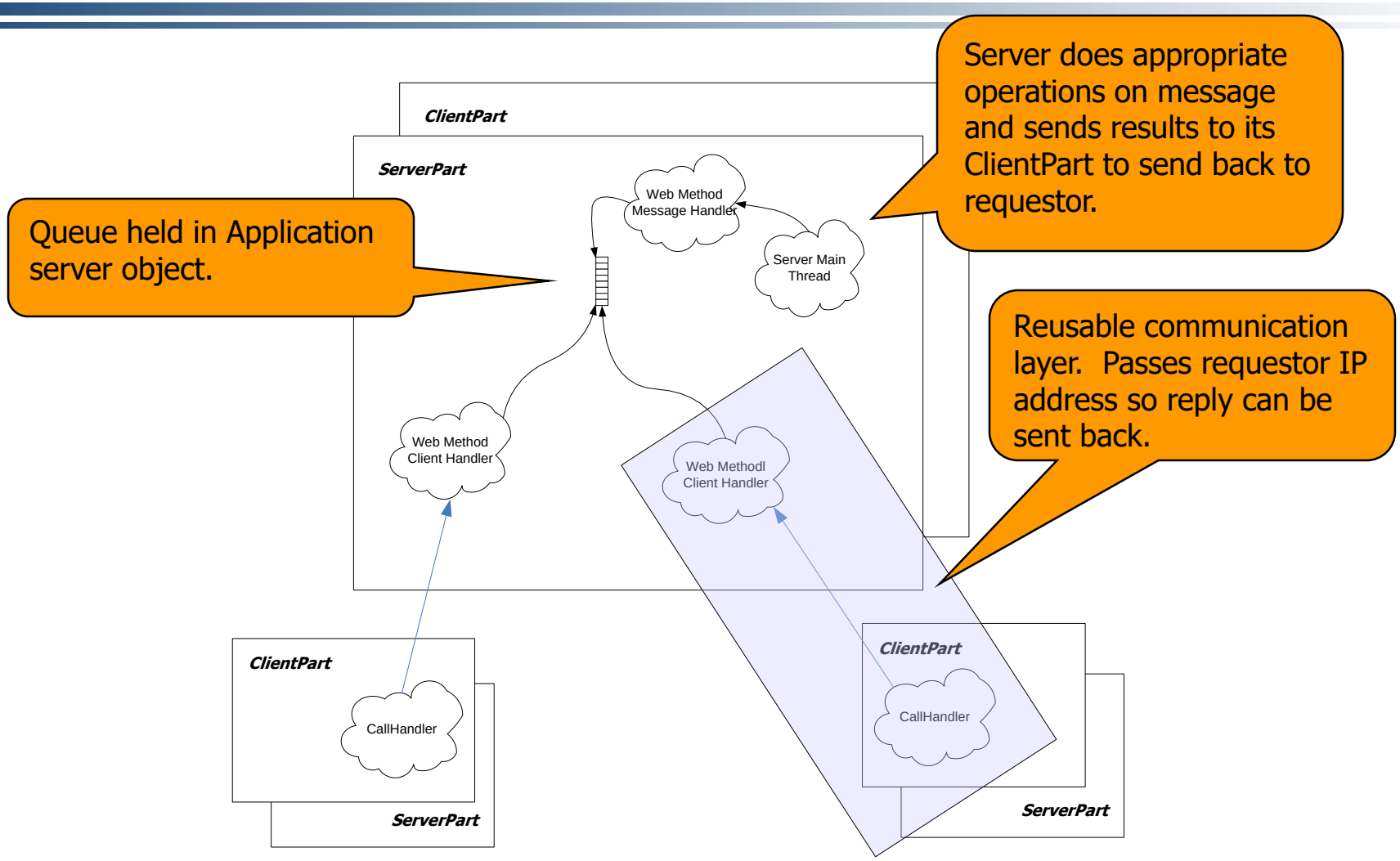
Abstract Factory Pattern



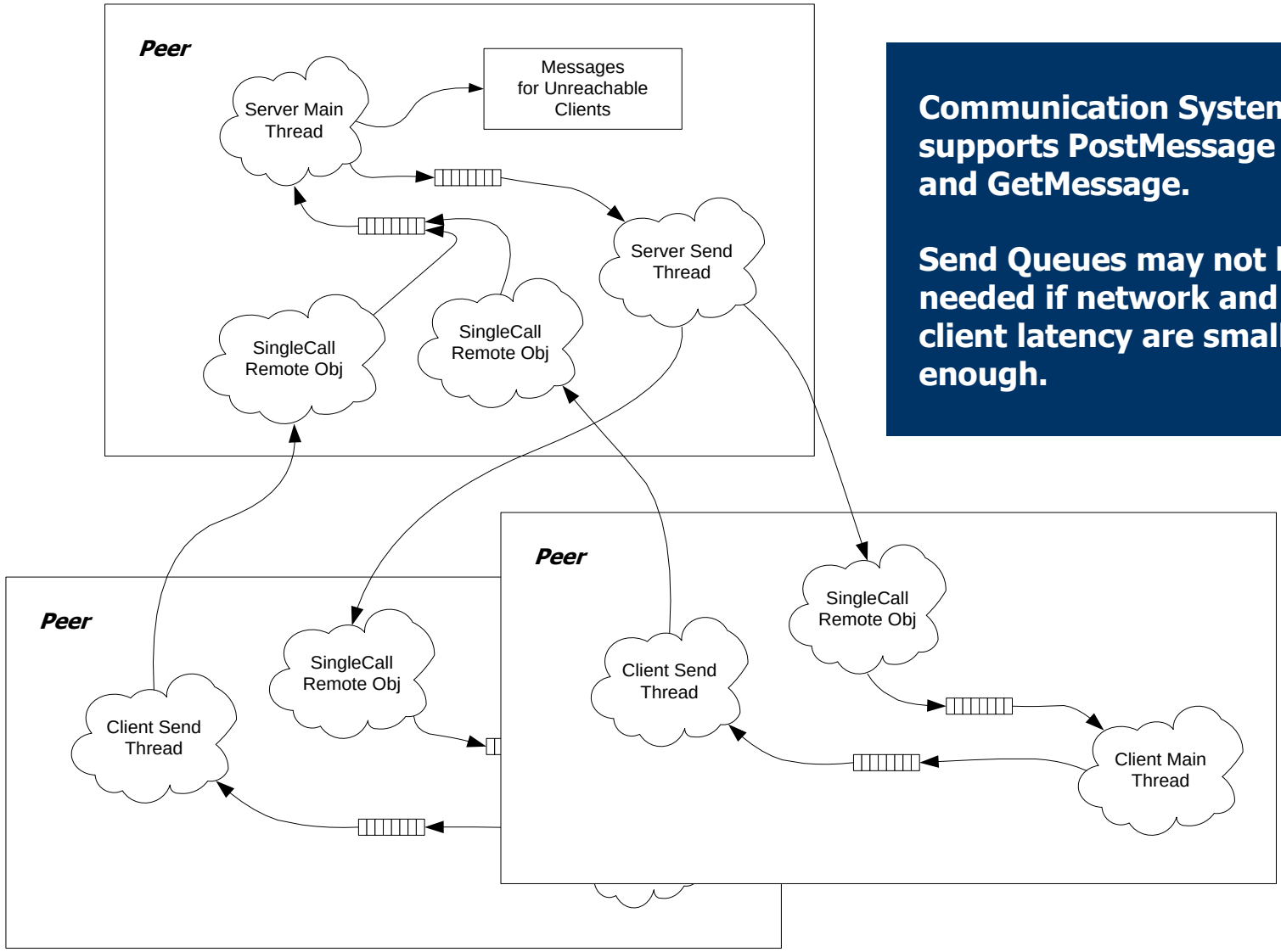
A Candidate Remoting Architecture



Candidate Web Services Architecture



Message Passing Communications



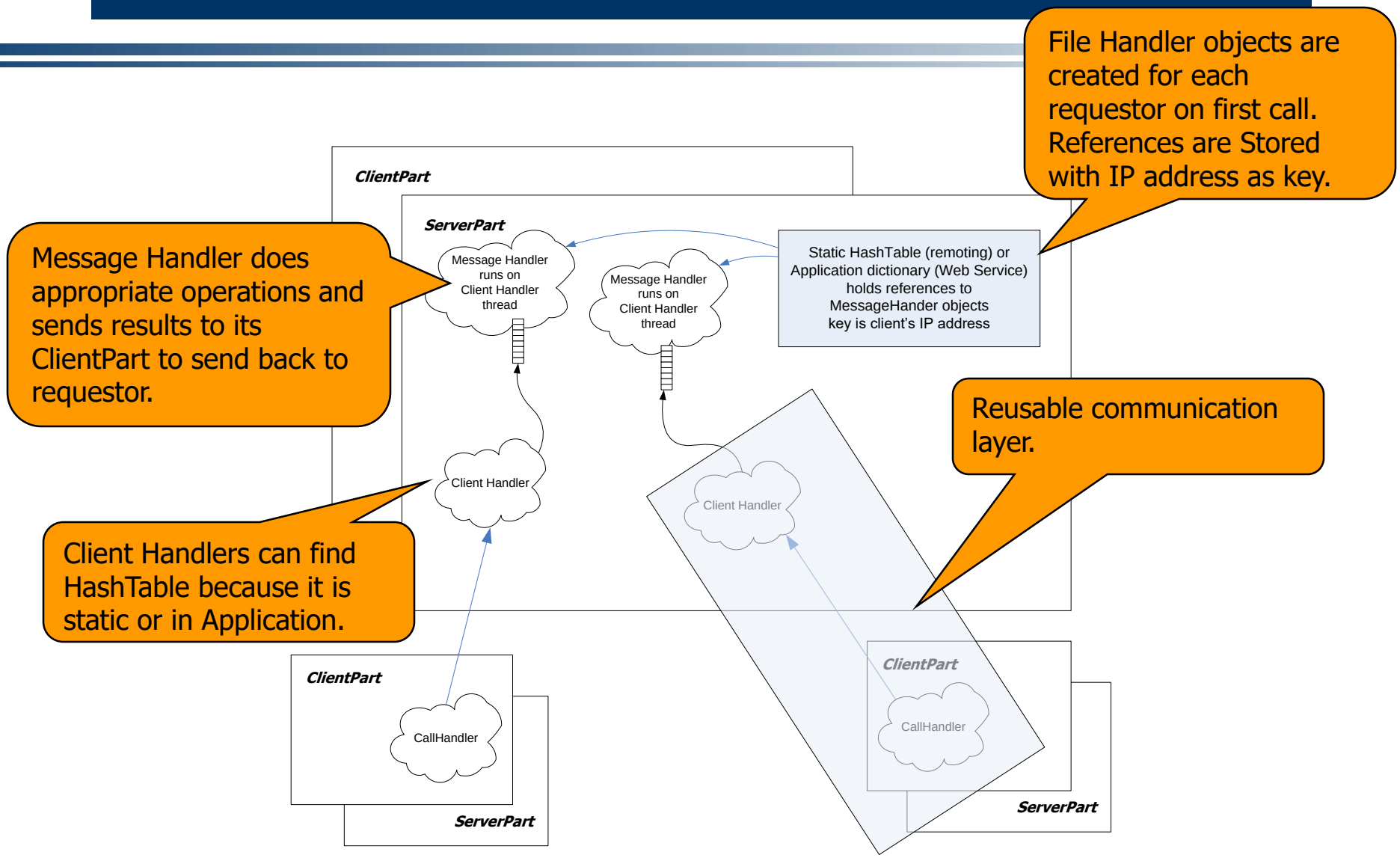
Communication System supports PostMessage and GetMessage.

Send Queues may not be needed if network and client latency are small enough.

Rationale

- This design factors server processing into:
  - Client Handlers that are independent of other clients, used to post to a single server queue, made available through a static function.
  - A central server part, running on the server's main thread.
- Use of server main thread input queue requires the use of messages.
  - Calls on the main thread are now deferred until the server gets around to servicing them.
  - Processing is serialized based on messages using a first-come-first-serve order imposed by the queue.
- This structure is appropriate if clients have to share server state (that's not the same thing as sharing files).
- The communication channel can be reusable because Client Handlers do no application specific processing. They just enqueue messages to the core server part.

Modified Candidate Server Architecture



Rationale

- This design factors server processing into:
  - Client Handlers are independent of other clients, used to post to a client specific queue held in static map or web service Application.
  - Message Handlers that are independent of other client requests.
  - We will still pass messages to the Server Part. Doing so helps us keep the communication layer reusable, as all the Client Handler has to do is to post client messages to its queue. The Message Handlers are application specific, but are made available through abstract factory and abstract application interfaces.
  - Processing now is fairer, allowing small requests to be served quickly rather than sitting waiting in a central queue for large requests to complete.
- This structure is appropriate if clients don't have to share server state (that's not the same thing as sharing files).

Message-Passing

- All communication is via messages rather than function calls.
  - Messages are sent via a PostMessage(msg) function call.
    - but its purpose is to send the message
    - It does not expect a return value nor does it wait for processing to complete.
  - Return values, for this Peer-To-Peer architecture, are sent back over a separate channel to the requestor's ServerPart.
  - Server-side processing is determined by message contents, not by which function is called.
  - Everything is serialized by first-come-first-serve queues.
  - All messages are formed as XML strings.
  - File transfer is effected by using Base64 Encoding to convert a binary file into a text string.

Consequences

- Client/server interface becomes much simpler – it just provides message passing.
- Now we have to add message parsing and processing.
  - A lot of the parsing can be handled by the XML DOM.
- Client and Server are not bound to each other in time.
  - Message passing works like email.
    - You can send a message whether or not the recipient is listening.

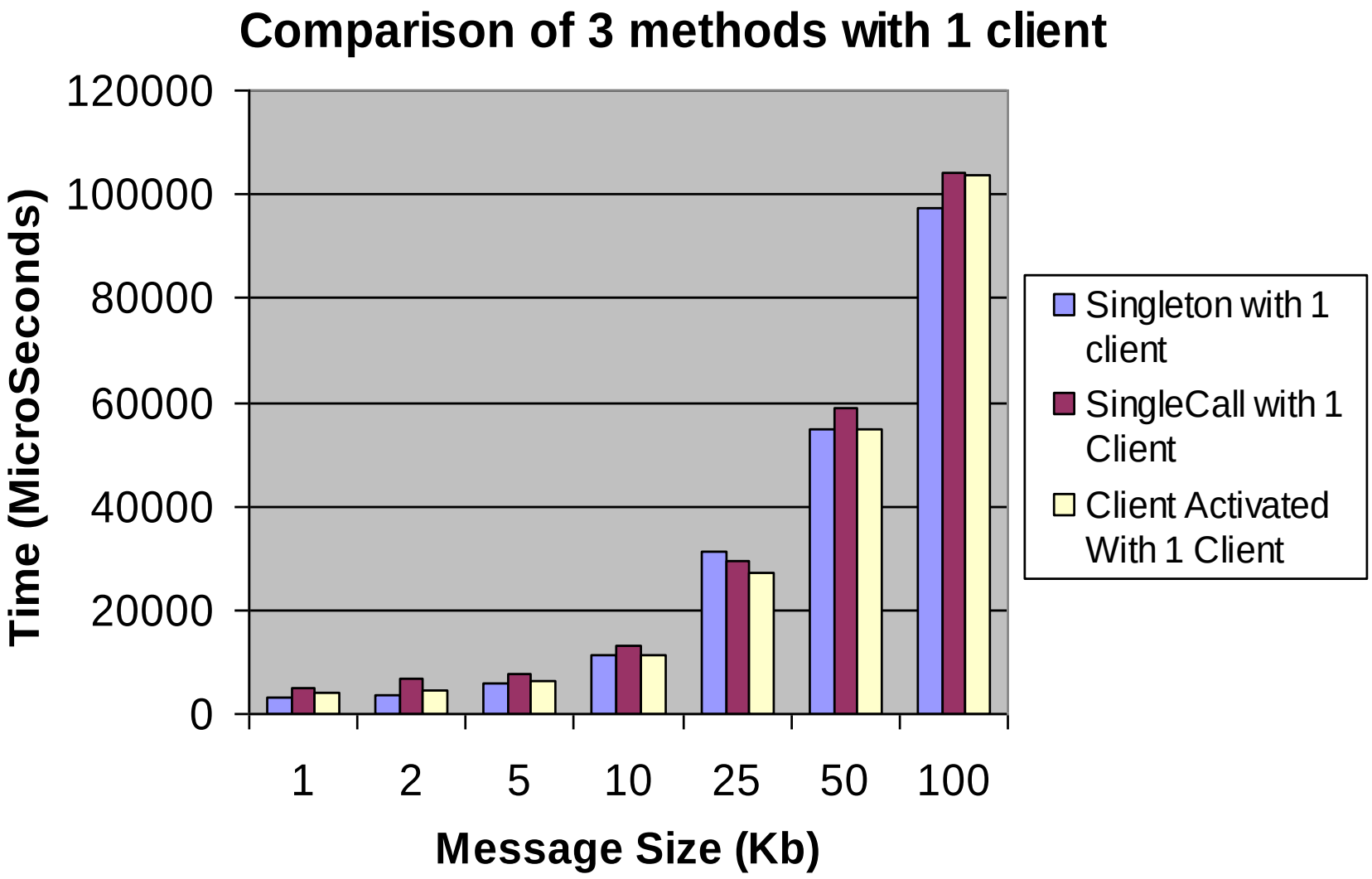
Advantages of Message Passing

- Client and server are decoupled in time.
  - Messages are serialized in receive queue(s) and so it is easier to manage server processing.
  - Server can send notifications whenever appropriate.
    - If client is running, notifications come whenever server sends them.
    - Otherwise, client simply collects pending messages when connecting to the server.
  - For file downloads, client can break connection and allow the server to send the files whenever it is ready.
    - Frees up resources on server and on the client.
- Communication is more reliable.
  - If clients are not available, server simply stores messages and lets client collect them later.
- Communications can be factored into a module and reused.

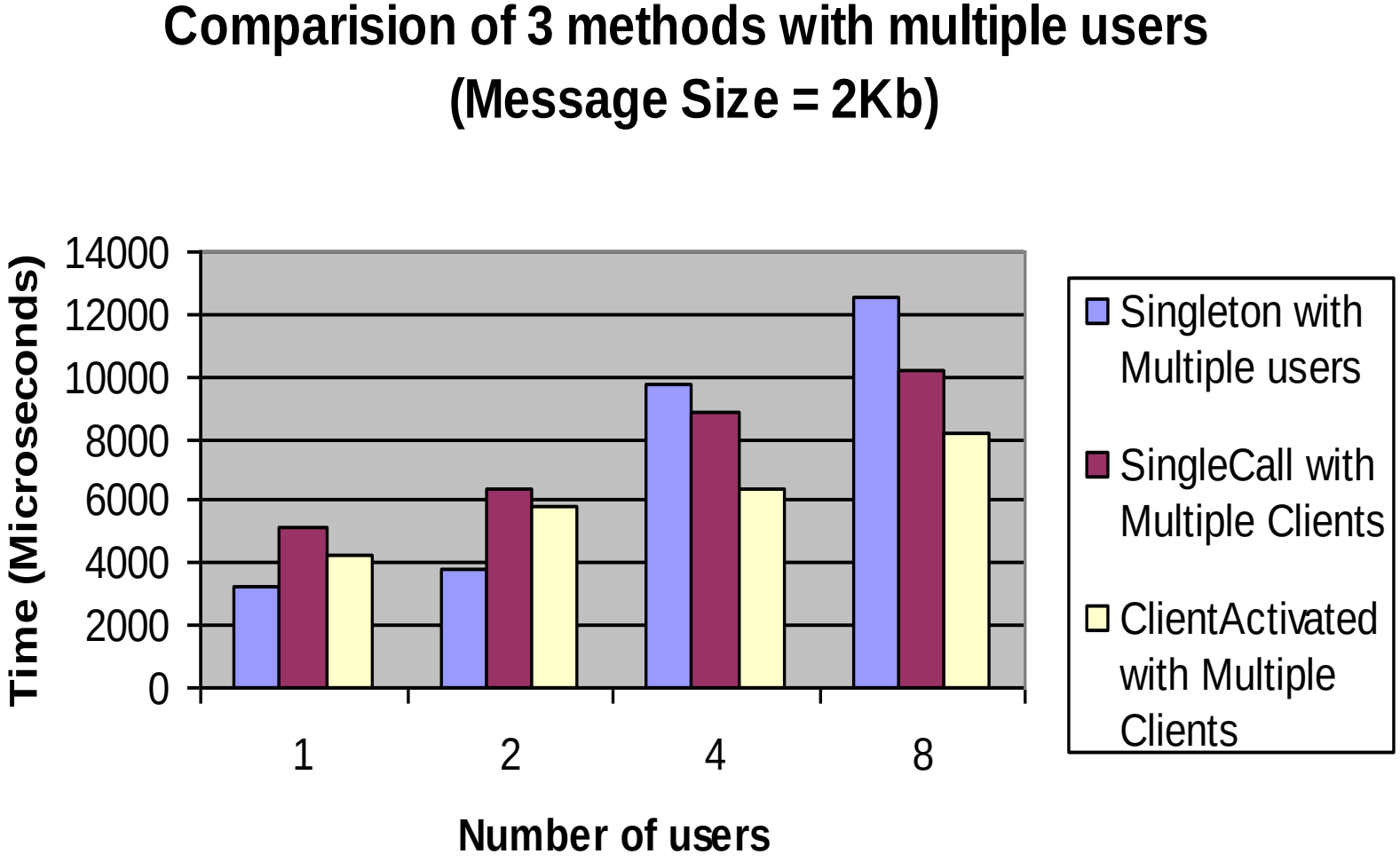
Performance Data

- On the next two slides find performance data taken for a message-passing system configured just like the previous page.
  - Measurements taken by Poonam Bijlani for her Final Project for CSE775 – Distributed Objects, Spring 2003.
- You may want to take performance data like this from your own prototype.
  - If you are running out of time, you can extrapolate from this data and describe how you would measure from a prototype if you had more time.

Remoting Performance: Activation vs Message Size



Remoting Performance: Activation vs Number of Users



End of Presentation