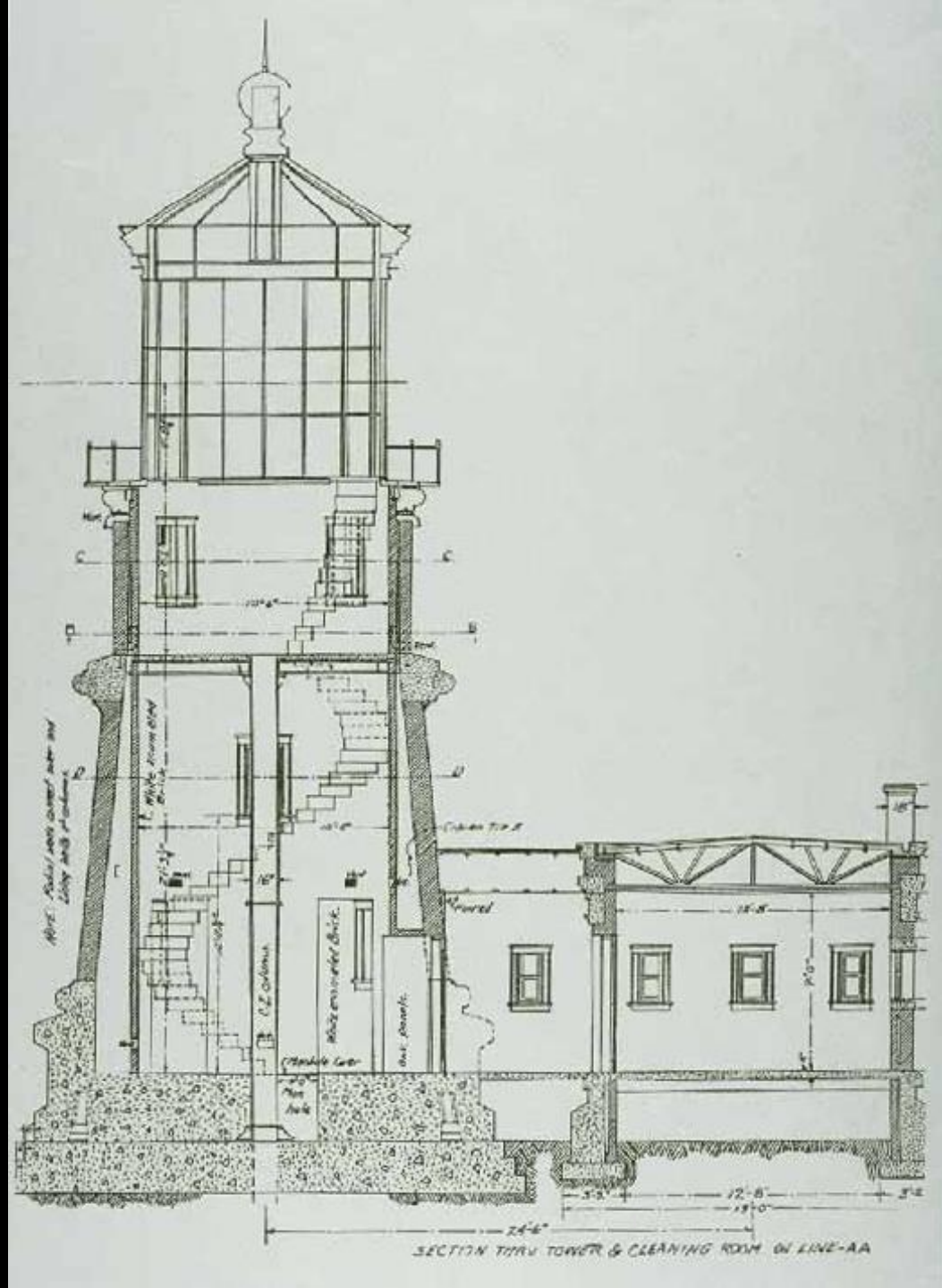
Software Architecture & Design



Middleware

The ‘slash’ in Client/Server

Contents

- Introduction
- History & Context: How middleware emerged as a new discipline.
- Interprocess Communication (IPC)
- Types of Middleware
- Anatomy & Mechanics of IPC
- Discussion & Q&A

History & Context

How ‘Middleware’ emerged as a new technology and discipline

Context & History: PCs & Networks

- Early Adopters: Large Industry circa 1960's
  - Mainframes – Highly Centralized
  - Terminal Connections (i.e. dumb terminals)
  - e.g. Large Banks & Telecoms
- Mini Computers
  - Smaller Enterprises & Universities
  - E.g. AS/400, VAX

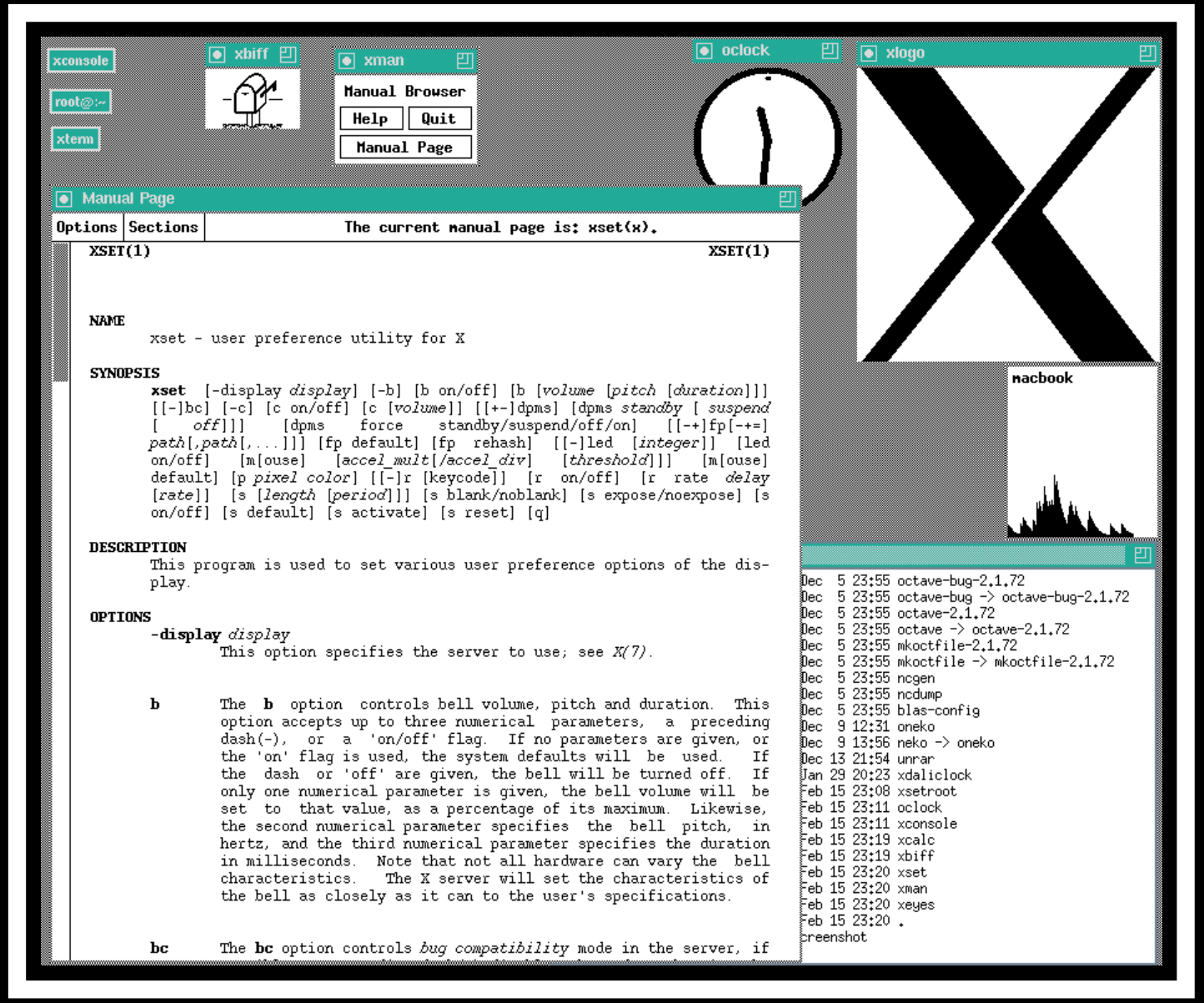
IBM 7090

Mainframe Computer

https://www.pinterest.ie/pin/483503709964490460/

Context & History: PCs & Networks

- Unix – Circa 1970's
  - Multitasking, Multi-user operating system (OS)
  - Bell Labs (AT&T)
    - Denis Ritchie, Ken Thompson, Brian Kernighan
    - Picked up by Universities and other corporates
  - X-Terminals
    - Graphic Terminals providing Windowing – i.e. Reverse Client/Server
    - WIMP – Windows, Icons, Mouse & Pointer
- The PC - Personal Computer – Circa 1980's
  - Apple II - 1977
  - IBM PC



X-Windows

X11 -Windowing System

By Liberal Classic - Liberal Classic, MIT, https://commons.wikimedia.org/w/index.php?curid=1680280

IBM PC

IBM 5150 PC

Apple PC

Apple IIe

https://commons.wikimedia.org/wiki/File:Apple_II_tranparent_800.png

Decentralization of IT Systems

- PC's become quickly adopted
  - 'Killer Apps' – e.g. Lotus 123
  - Data being stored outside IT
- Servers & LAN Technology Emerges
  - Token Ring, Ethernet, Token Bus
  - Need to share resources (e.g. File & Print) – Economics in 80's and 90's forced 'sharing'
    - Large storage disks were still (relatively) expensive
    - Printers were expensive

Client/Server Revolution

Hardware Costs

- Mainframes & Minicomputers are expensive – really expensive
- PC-Servers are relatively inexpensive
  - $5,000 versus $5,000,000
- Capacity increases smoothed out
- Reduced Risk
- MIPS Comparisons*

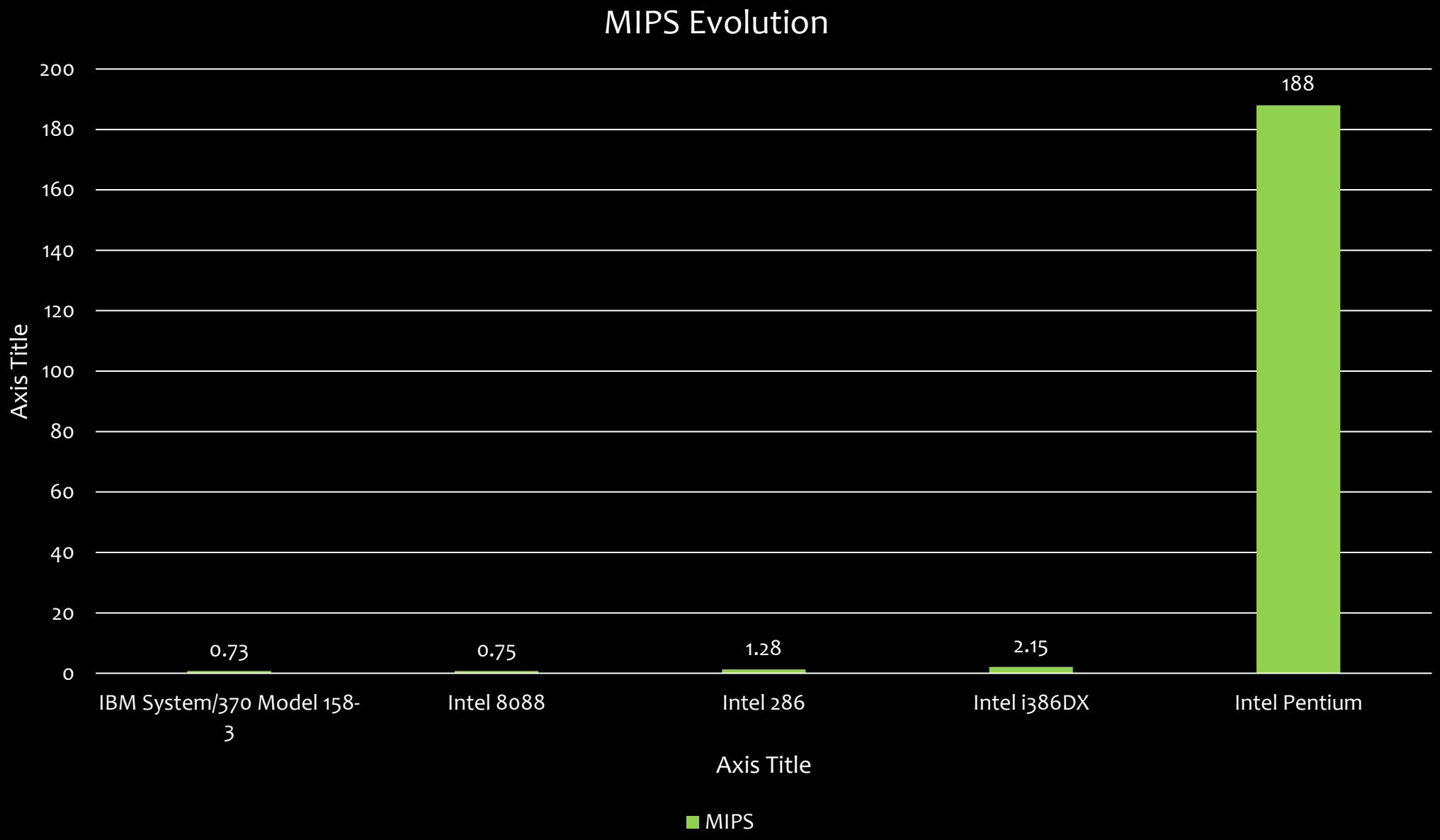
Development Costs

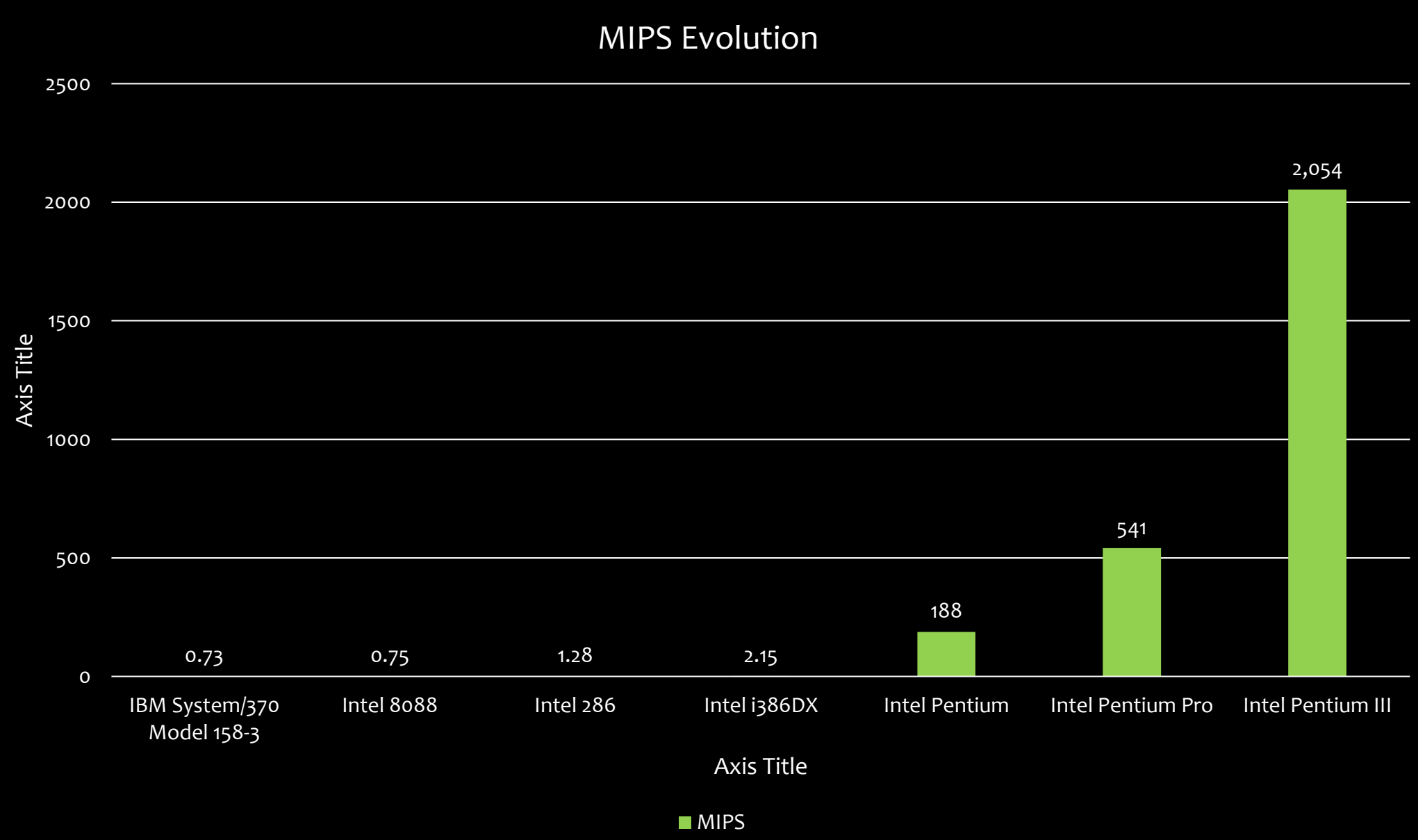
- New development tools emerge
  - PC-era Software Houses start to dominate
- Becomes cheaper to develop new systems
- Becomes faster to develop new systems
- Business Unit autonomy

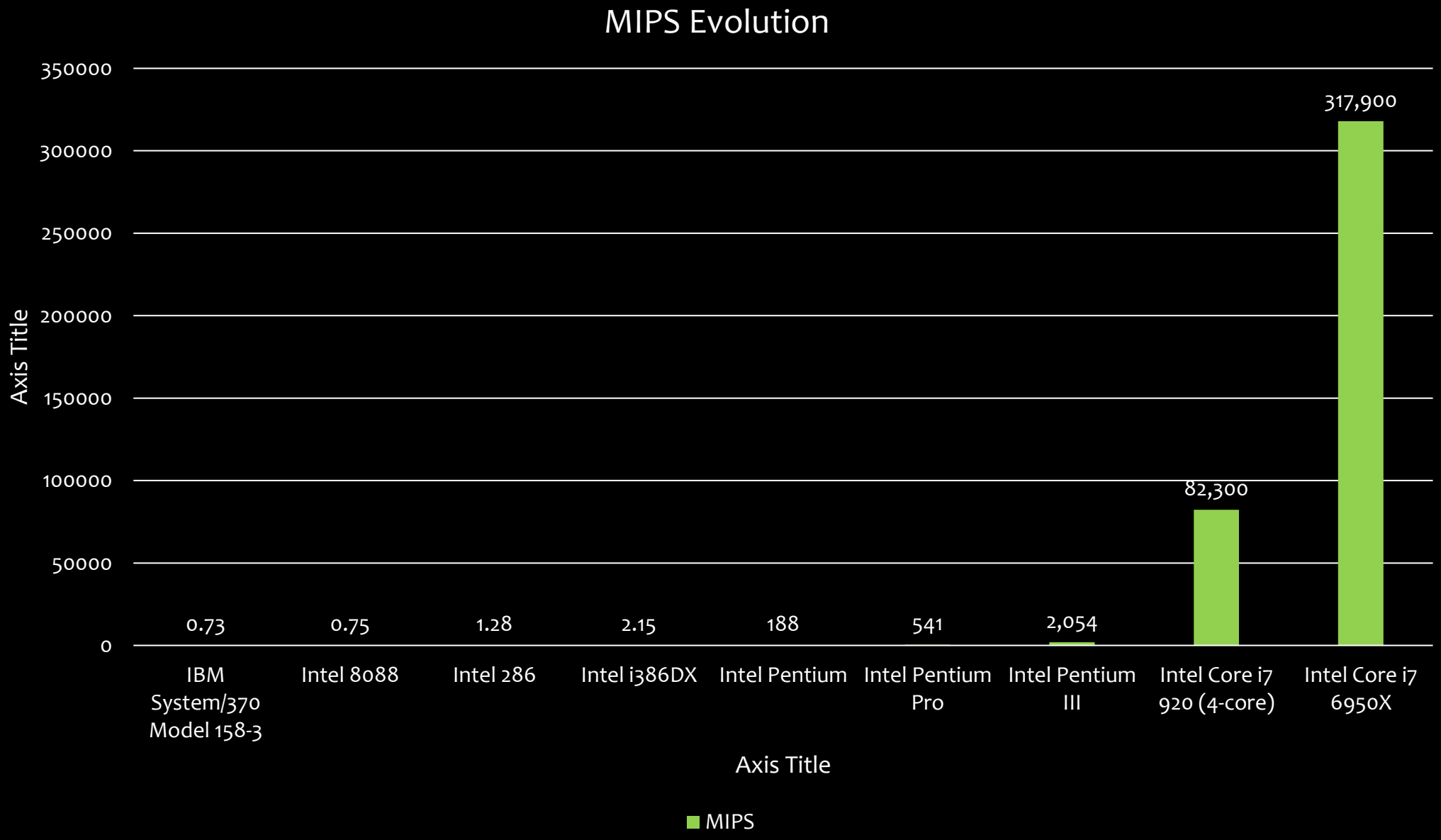
MIPS Comparison

Processor	MIPS	Year
IBM System/370 Model 158-3	0.73 MIPS at 8.696 MHz	1977
Intel 8088	0.75 MIPS at 10 MHz	1979
Intel 286	1.28 MIPS at 12 MHz	1982
Intel i386DX	2.15 MIPS at 16 MHz	1985
Intel Pentium	188 MIPS at 100 MHz	1994
Intel Pentium Pro	541 MIPS at 200 MHz	1996
Intel Pentium III	2,054 MIPS at 600 MHz	1999
Intel Core i7 920 (4-core)	82,300 MIPS at 2.93 GHz	2008
Intel Core i7 6950X	317,900 MIPS at 3.0 GHz	2016

Source: https://en.wikipedia.org/wiki/Instructions_per_second#MIPS







Case Study

NatWest Bank – Retail Banking Platform (mid ‘90s)

Interprocess Communication

Communicating Across the Process Boundary

Basic IPC

- Signals
  - Basic interrupt
  - Unix/Linux ‘kill’ command
- Pipes
  - Supports IPC between related processes
  - Unidirectional communication mechanism – e.g. `$ ls -ln | grep “filename”`
- Named-Pipes/FIFO
  - Supports IPC between unrelated processes
  - Special Pair of File Descriptors
  - Read/Write from File Descriptor

Basic IPC - Sockets

- Sockets
  - Unix Domain Sockets (IPC Socket)
  - Network Sockets & TCP/IP (aka Berkley Sockets)
    - DARPA funded TCP/IP & Berkley Unix (BSD)
- Formal Roles of ‘client’ and ‘server’
  - Send/Receive – i.e. Full Duplex Streaming Communication Channel

RPC – Solving the Network Issues

- Remote Procedure Call (RPC)
- Preservation of ‘Procedure’ programming paradigm
- IDL – Interface Definition Language
- Marshalling/Unmarshalling
- Encapsulate low-level network socket programming

CORBA – Object Oriented RPC

- CORBA - Object Management Group
- ORB – Object Request Broker
- IIOP – Interoperable protocol (adapted for TCP/IP)
- CORBA Services - Enterprise Services
  - e.g
    - Name Service/Binder
    - Event/Notification Service
    - Object Transaction Service

Web/Internet

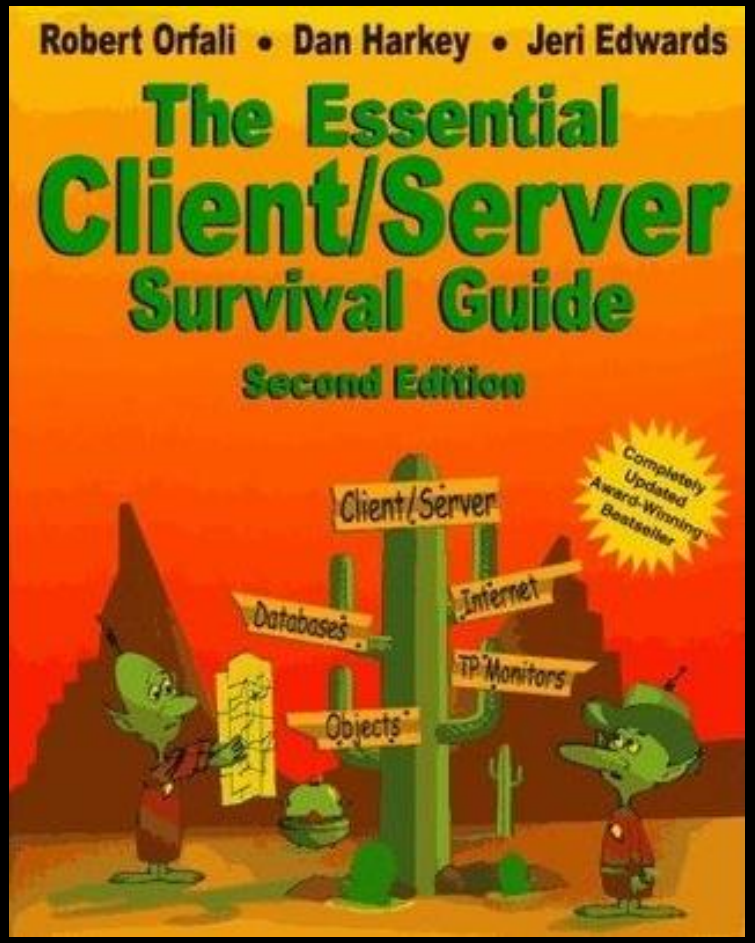
- HTTP
- Firewalls
- Desktop & OS Wars
- Adoption of browser-based applications
- SOAP (Simple Object Access Protocol)
- Servlets
- WebServices
- XML/RPC
- XML & JSON
- REST

Types of Middleware

Broad Classification

Types of Middleware

- SQL
- TP Monitors (Teleprocessors)
- Message-Oriented-Middleware (MOM)
- Groupware
- Distributed Objects
- Internet/HTTP-based



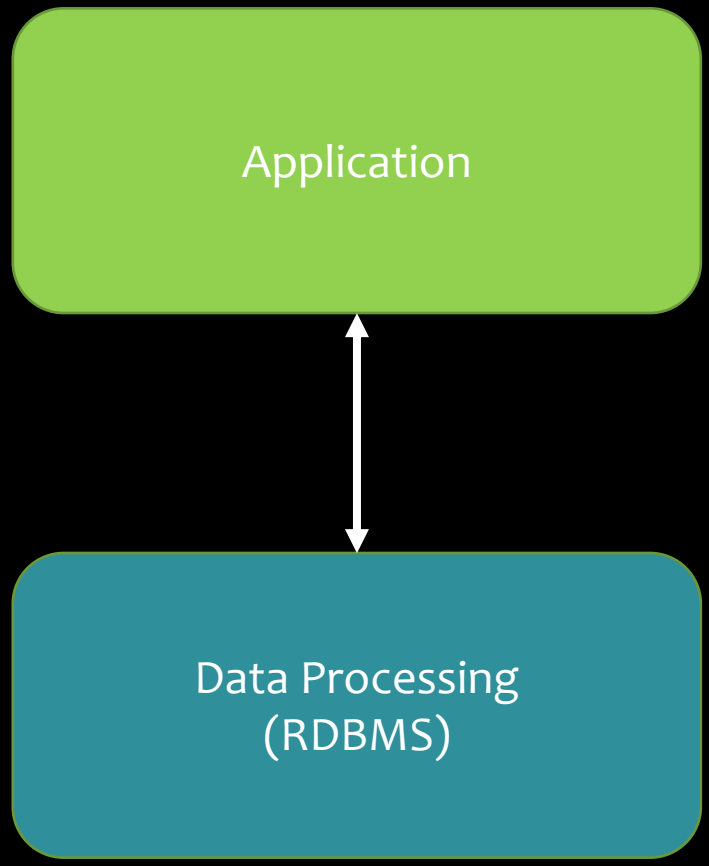
SQL

- SQL Requests issued to the RDBMS Server
- Result ‘set’ returned to client
- Abstracts the network programming
- Abstracts how the data is accessed
- SQL Server Engine 3-Layer Model
  - External Schema
  - Internal Schema
  - Physical Schema

SQL – 2 Tier Model

1. Application
2. Data Processing

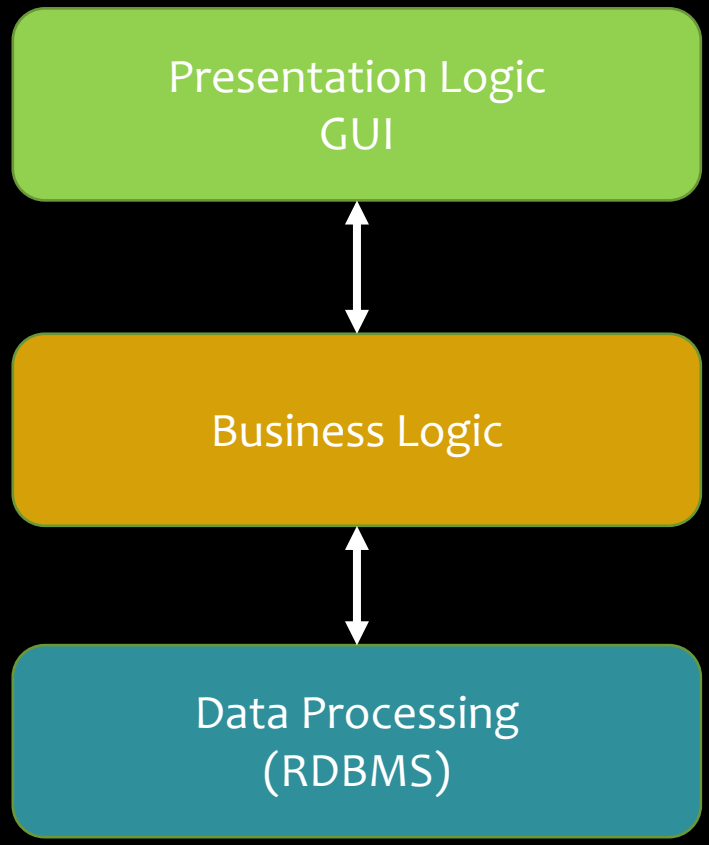
- E.g.
  - MS Visual Basic & MS SQL-Server
  - Pro C & Oracle



SQL – 3 Tier Model

1. Presentation Layer
2. Business Logic
3. Data Processing

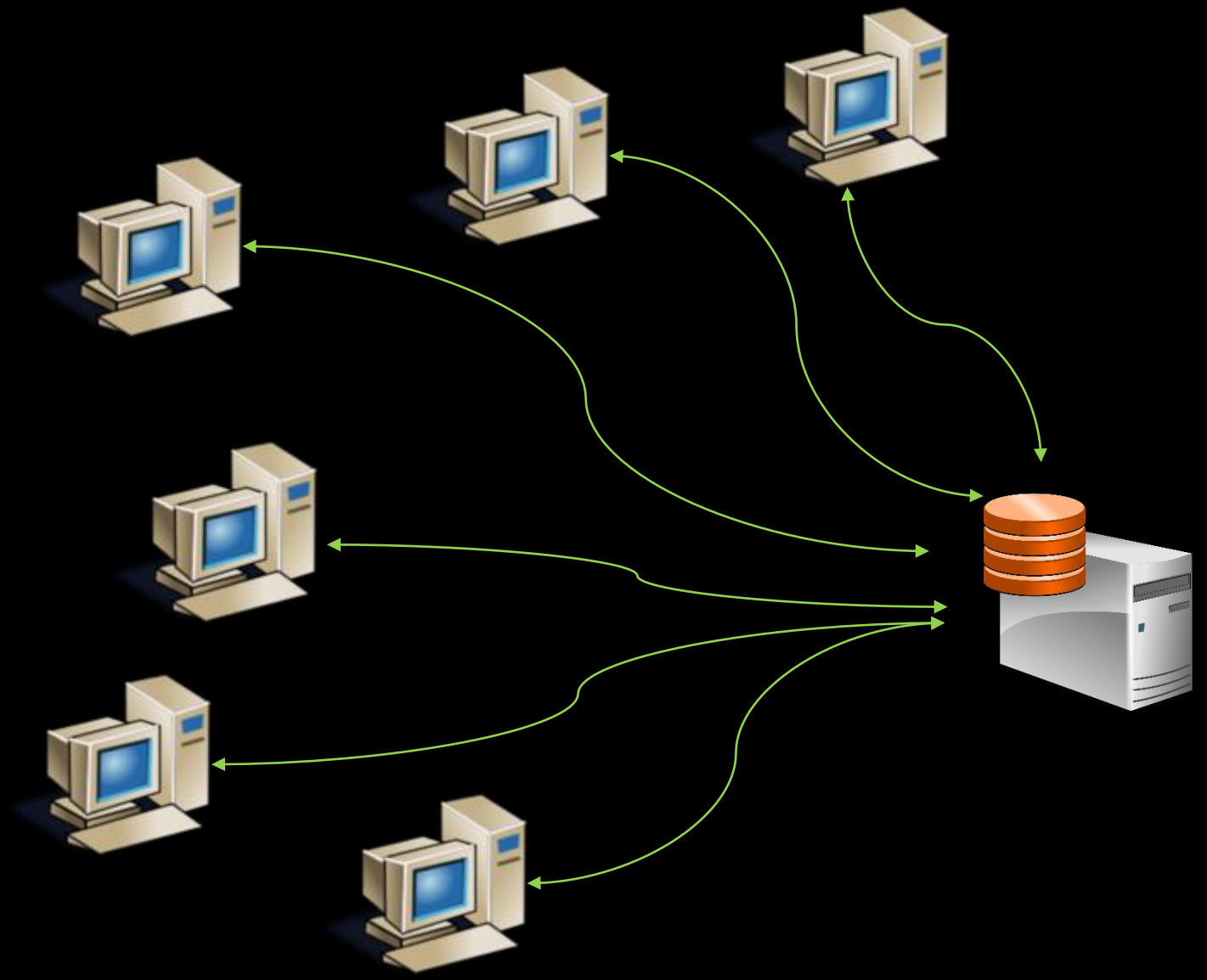
- Example Stack:
  - MS Visual Basic client
  - Visual C++ Server
  - MS SQL-Server

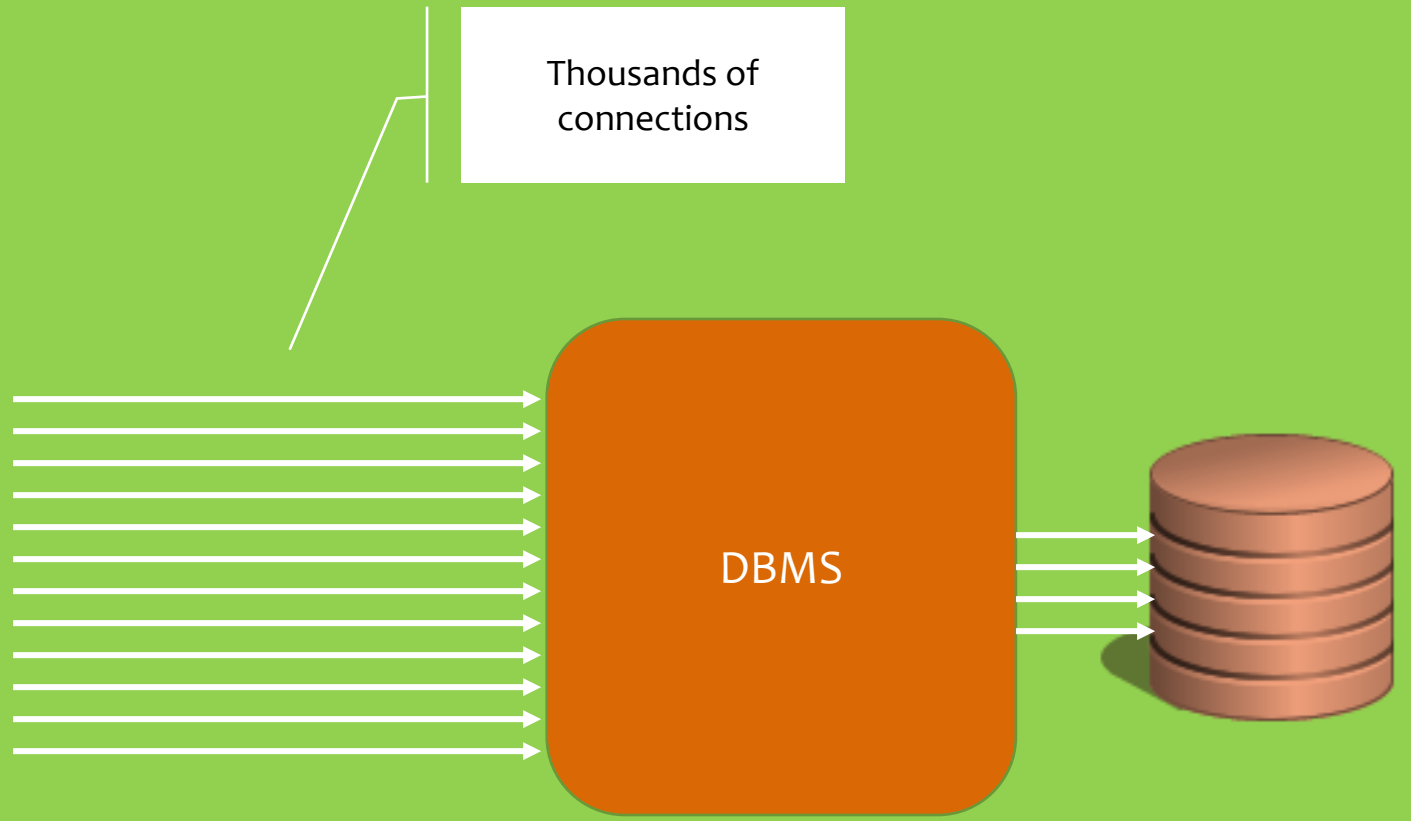


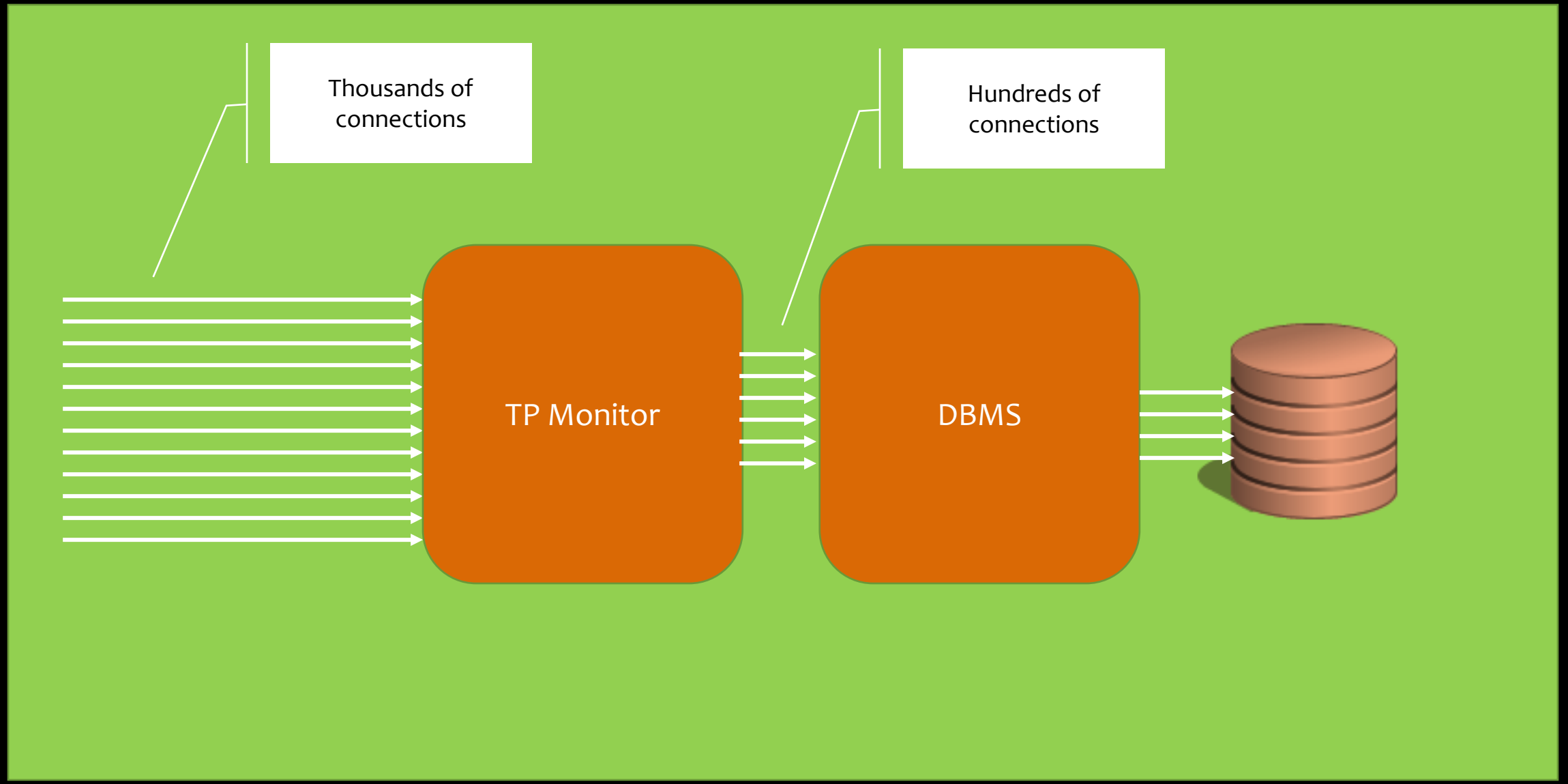
TP Monitors

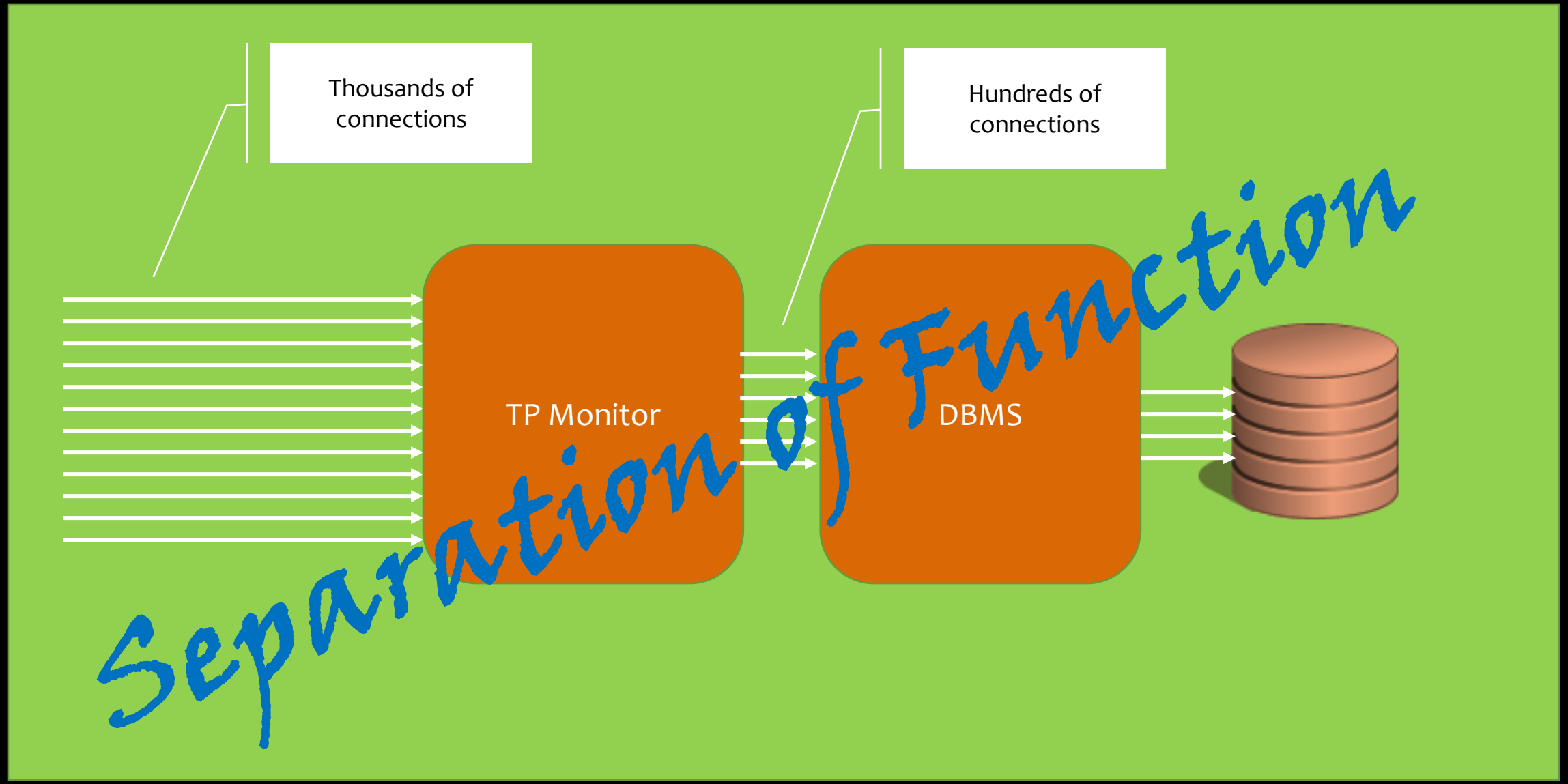
- TP Monitors/Teleprocessing Monitors
  - Allow many clients to connect to one server
  - Manages transfer to multiple requests to a server
- Transaction Processing Monitors
  - Manages Transactions
  - Co-ordinates Distributed Transactions (across many databases)
  - Handles thousands of txns per second

- E.g.
  - IBM CICS (Customer Information Control System)
  - Oracle Tuxedo









Message Oriented Middleware

Characteristics

- One-way Message
  - Not Request/Response
- Fire-and-Forget
- Asynchronous, non-blocking
- Connectionless
  - Less expensive
- Often infers queuing

Benefits

- Scalability
- Lends itself to 'Event Model' architecture
- Favored in Microservices Architecture
- Popular in IoT

Groupware – Collaborative Software

- Workflow
- Remote Group Collaboration
- Social
- Collaborative Documents

Distributed Objects

Technologies

- COM/DCOM
- CORBA
- RMI
- EJB
- BlazeDS*
- Spring Beans (Remoting)

Benefits

- Object-Oriented Model
  - Encapsulation helps with complexity & heterogeneity in Distributed Systems
  - Polymorphism helps with generics
- Domain Modelling

Internet/HTTP

‘90s → Early 2000’s

- COM/DCOM
- CORBA
- JavaWeb
  - RMI
  - Servlets
  - EJB
- SOAP
- WebServices

Mid 2000’s → Mid 2010’s

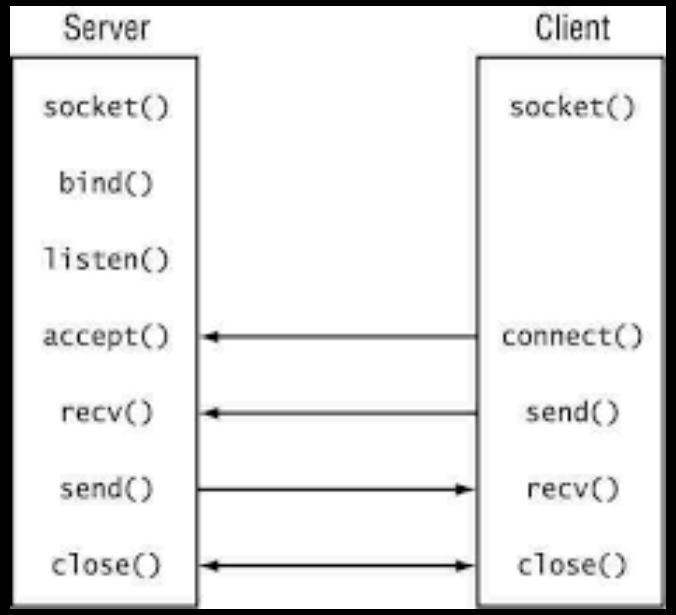
- Flex & BlazeDS/GraniteDS
  - ActionScript → Java
  - AMF – Adobe Messaging Format
  - RPC-like functionality
- JavaScript & JSON
- REST

Anatomy & Mechanics of IPC

Network Programming

Sockets

- Formal Roles of Client & Server
- Server-side Functions
  - Create Socket
  - Name Socket
  - Wait for Connections
- Client-side Functions
  - Connect
  - Send Data
  - Receive Data



https://stackoverflow.com/questions/36520590/basic-flow-of-control-in-socket-programming

Spot Question

What limitations or challenges arise with socket programming?

Network Sockets - Limitations

- Roles & Identity
  - Clarify Roles
  - Protocol & Request/Response Semantics
  - Structures needed to define Request and Input Data
  - Structures needed to define Response and Return/Output Data
  - Conventions required to Report Success/Failure or Otherwise
- Addressing
  - Host IP & Port Numbers
- Raw Stream – need for Type Conversion
  - Data Types – e.g. how big is an ‘int’?
  - 32-bit versus 64-bit
  - Endianness – i.e. big-endian or little-endian
- Timing
  - Race to start up

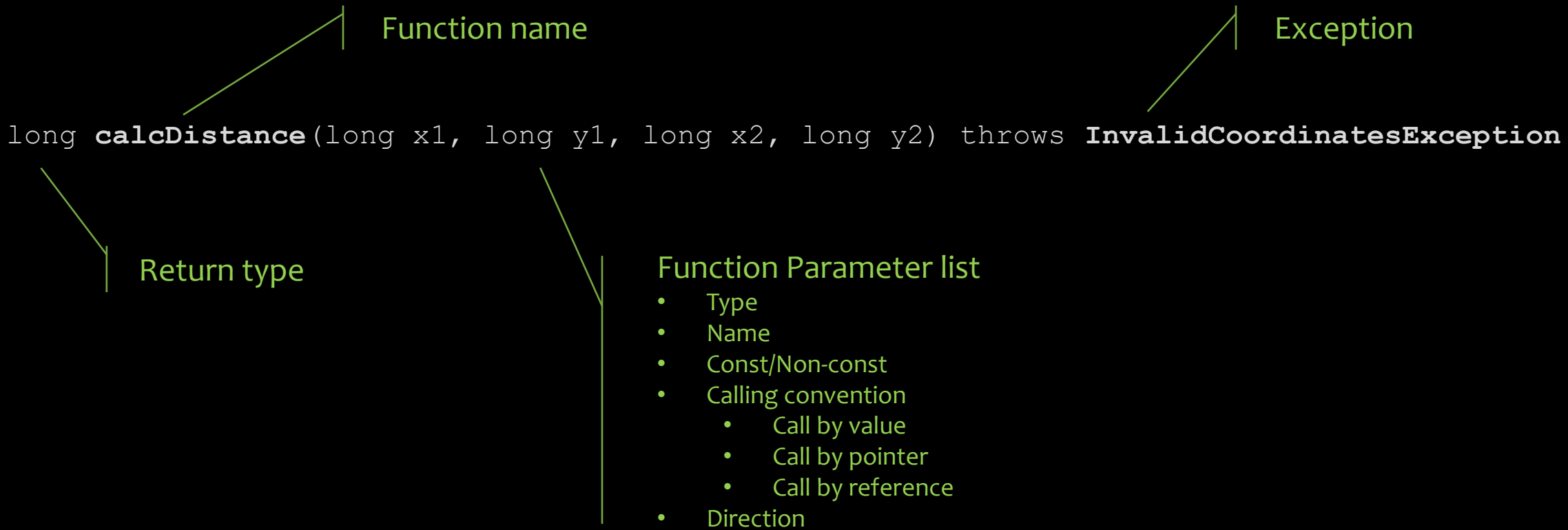
Socket Programming

Code Demo

Remote Procedure Call

- Preserve the ‘Procedure’ Paradigm
- Deconstruct function calls – i.e. signature
  - Function Name
  - Function Parameters
    - In, Out, InOut
  - Return type
  - Exceptions

Remote Procedure Call



Remote Procedure Call → OORPC

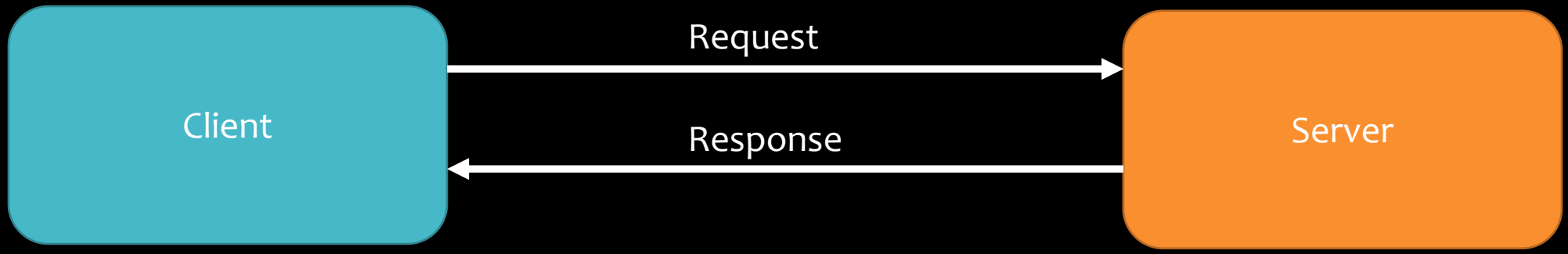
```
long calcDistance(long x1, long y1, long x2, long y2) throws InvalidCoordinatesException
```

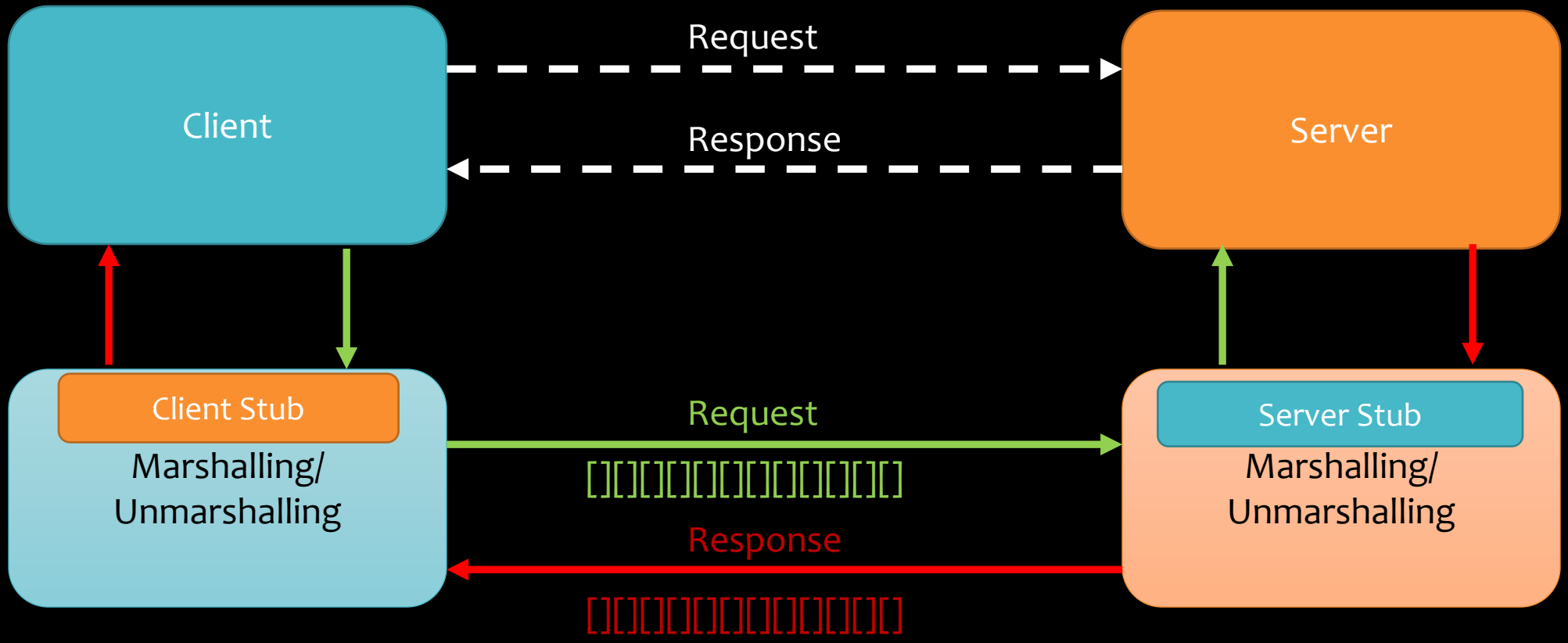
Class name (or possibly namespace)

```
long Planner::calcDistance(long x1, long y1, long x2, long y2)
```

```
long Acme::Planner::calcDistance(long x1, long y1, long x2, long y2)
```

Namespace (or possibly outer class name)





RPC versus LPC

- Performance
- Concurrency control v thread safe
- Reentrancy
- Non-deterministic v Deterministic

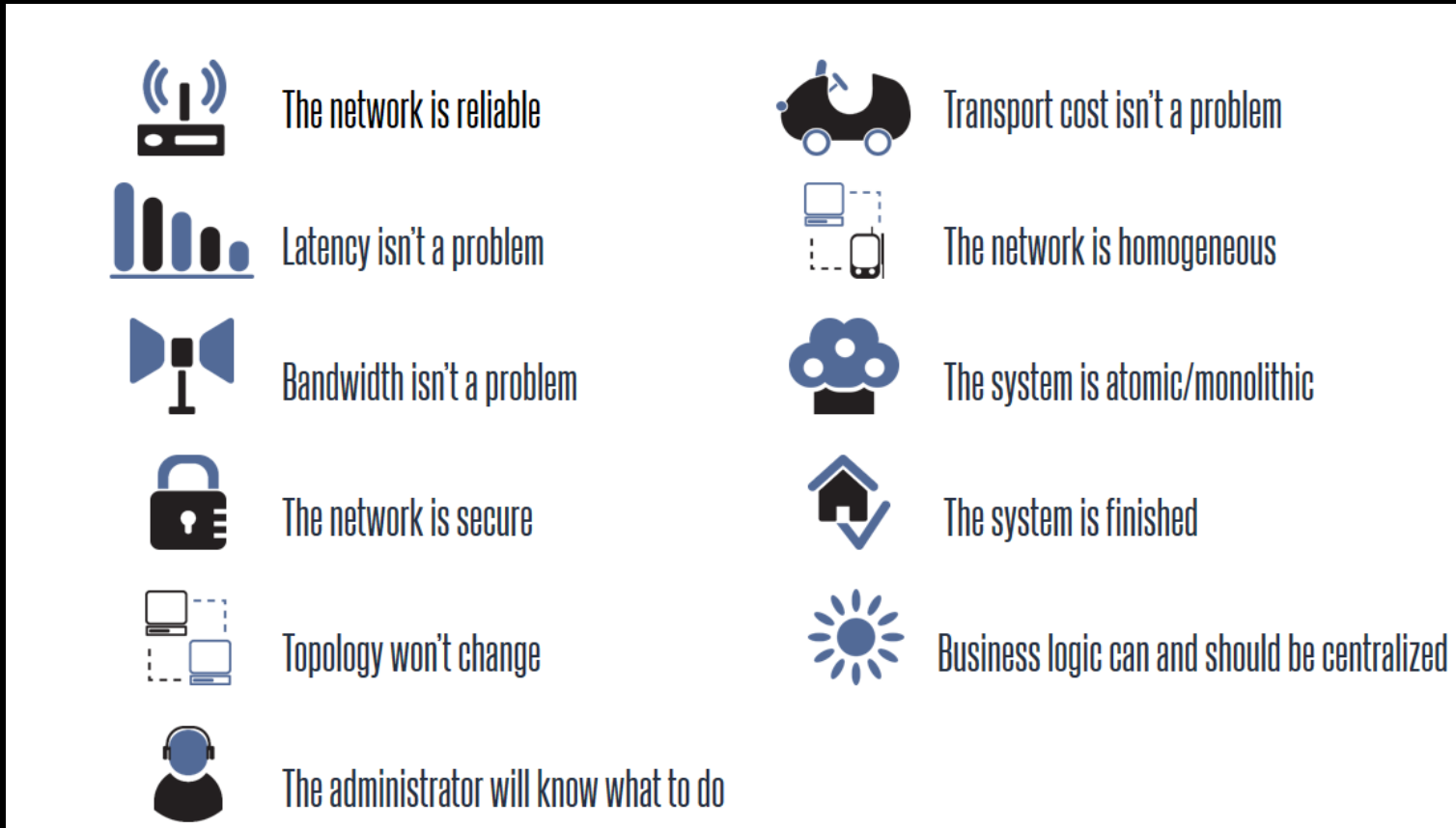
Q&A

Discussion Time

Summary

- Middleware is the most significant aspect of Client/Server
- Provides the mechanism and infrastructure to enable client/server and facilitate distributed systems development, including cloud.
- Various forms of middleware exist and the architect is responsible for selecting the technologies and where they get applied.
- Middleware is a stack and abstraction is a good thing

Fallacies of Distributed Computing



https://www.planetgeek.ch/2014/03/29/composite-ui-for-service-oriented-systems-fallacies-part-i/

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

The Eight Fallacies of Distributed Computing

Peter Deutsch

Essentially everyone, when they first build a distributed application, makes the following eight assumptions. All prove to be false in the long run and all cause *big* trouble and *painful* learning experiences.

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

For more details, read the article by Arnon Rotem-Gal-Oz

http://nighthacks.com/jag/res/Fallacies.html

Thank You

Recommended Reading

- Unix Network Programming *by W. Richard Stevens (Volume I and II)*
- Power Programming with RPC *by John Bloomer*
- Advanced Programming in the Unix Environment *W. Richard Stevens*
- Advanced CORBA Programming with C++ *by Michi Henning & Steve Vinoski*
- The Rise and Fall of CORBA *by Michi Henning*
- Building a RESTful Web Service – Spring Guides