

Reliable Transaction Router (RTR) Overview



Reliable Transaction Router (RTR)

RTR is a software fault tolerant transactional messaging middleware used to implement large distributed applications with client/server technology.

RTR Key Features

- Transactional middleware with recovery of in-flight transactions
- Distributed transaction manager which understands XA and DECdtm protocols
- Message Content based routing
- Horizontal scalability with data partitioning
- High availability with process and node redundancy
- Disaster tolerance capability with shadowed servers
- Programming API supporting:
 - C language bindings
 - C++ language Bindings
 - Java api (JRTR)
- Three Tier architecture

Reliable Transaction Router

Agenda

- Reliability and Fault-Tolerance
- Scalability
- Easy Building & Management
- Interoperability
- Web Applications
- To know more

Reliability and Fault-Tolerance

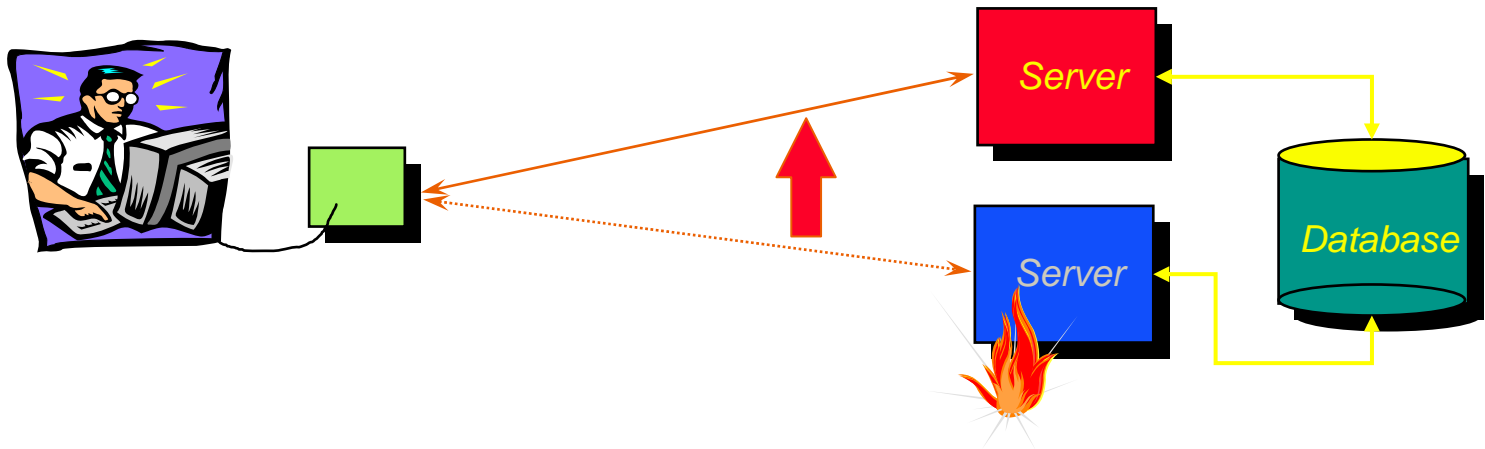


Traditional client-server (non-RTR example)



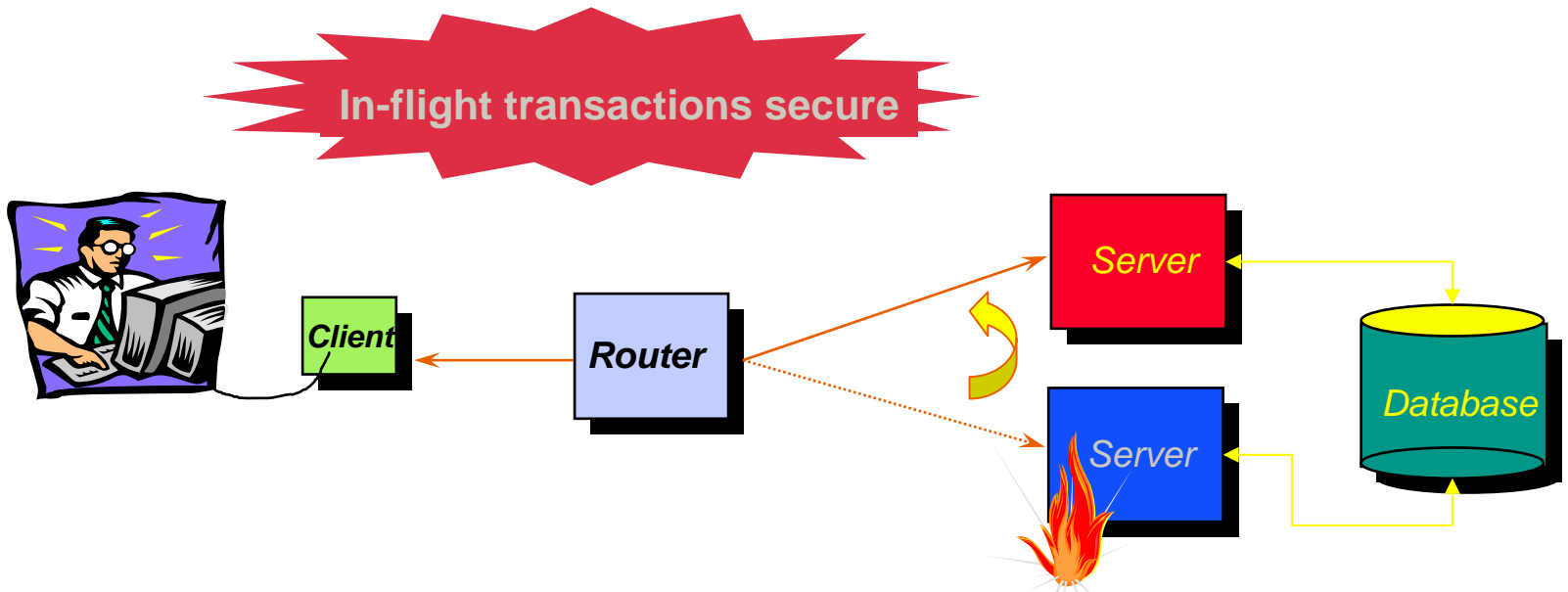
→ Servers can fail

Standby server (two tier model)



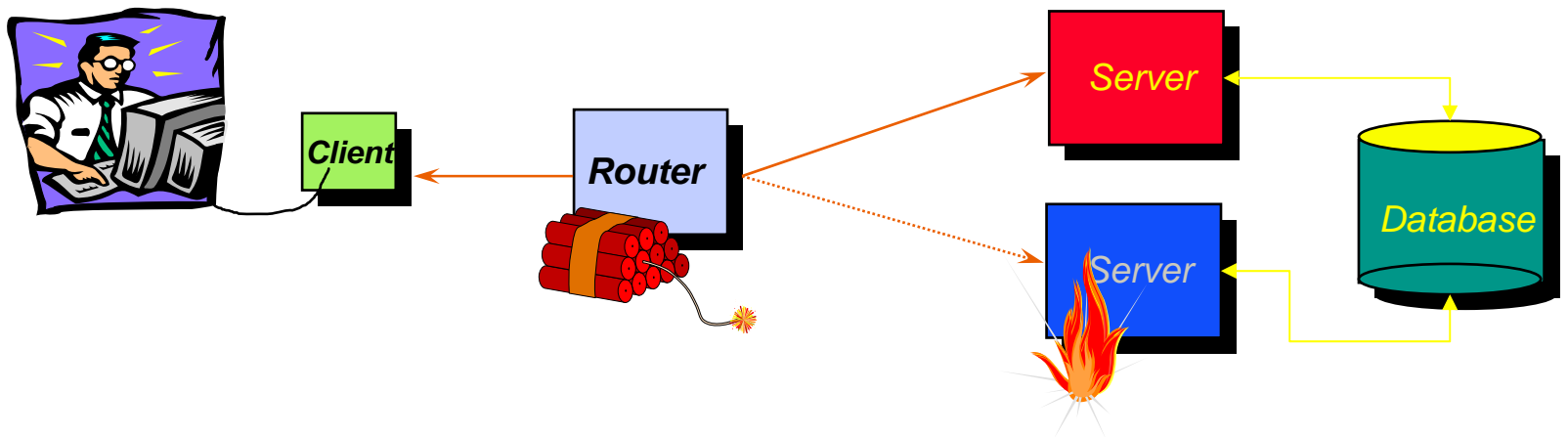
➔ **Client itself must fail over to a standby**

RTR 3-tier architecture



- ➔ **Server failover is transparent to client**
- ➔ **In-flight transactions are not lost**

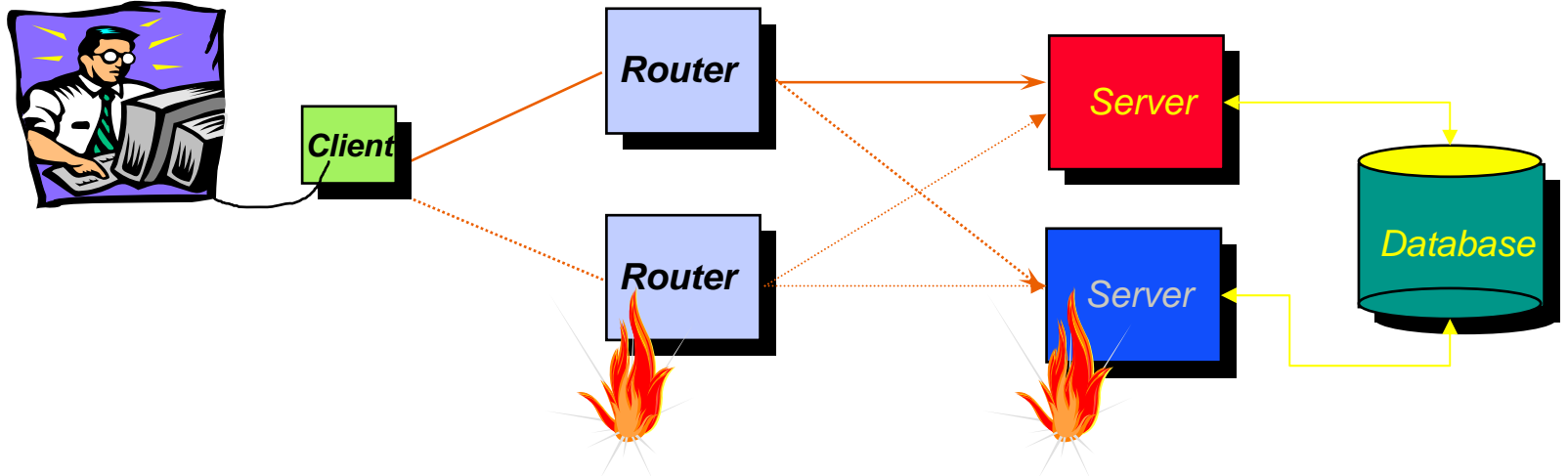
RTR 3-tier architecture



→ If a router fails

Standby router with RTR

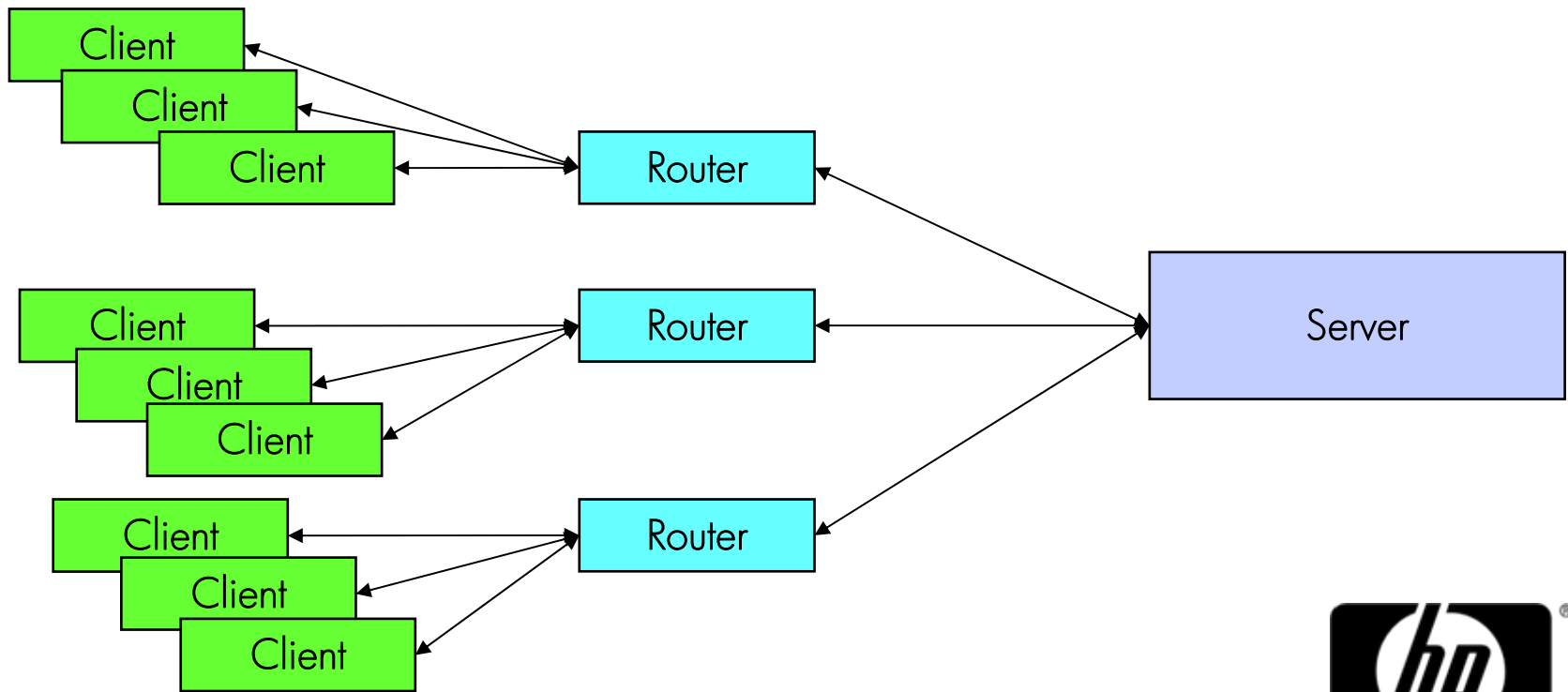
In-flight transactions secure



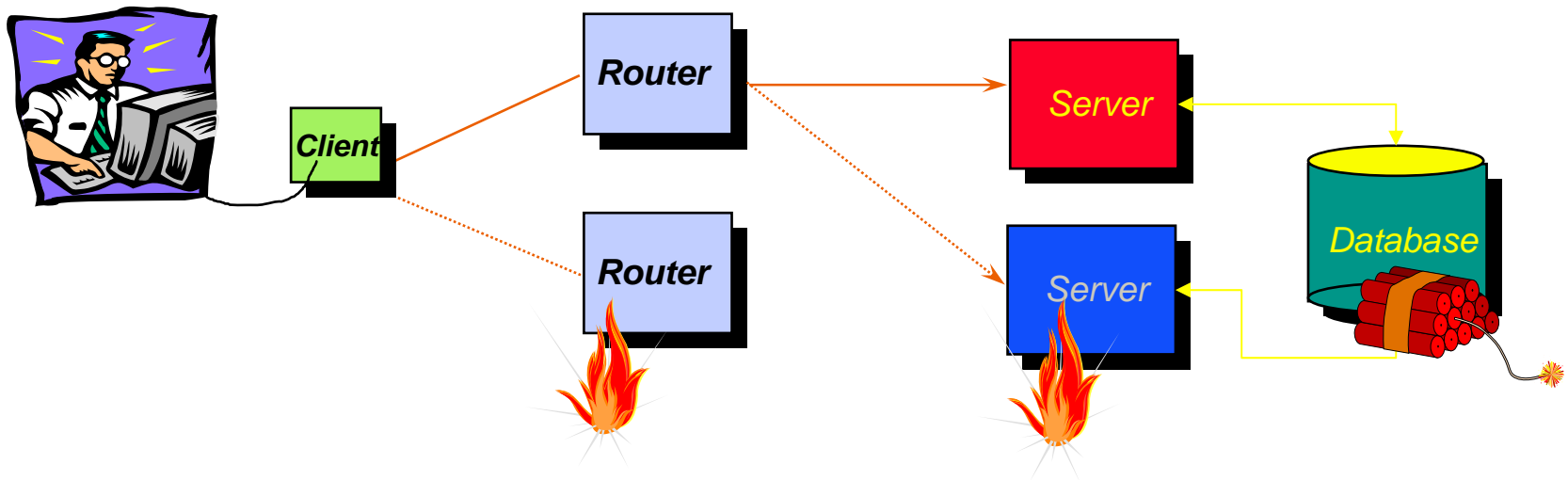
➔ Router failover is transparent to clients

RTR Three-tier Architecture

➔ **Client connection distribution across routers**

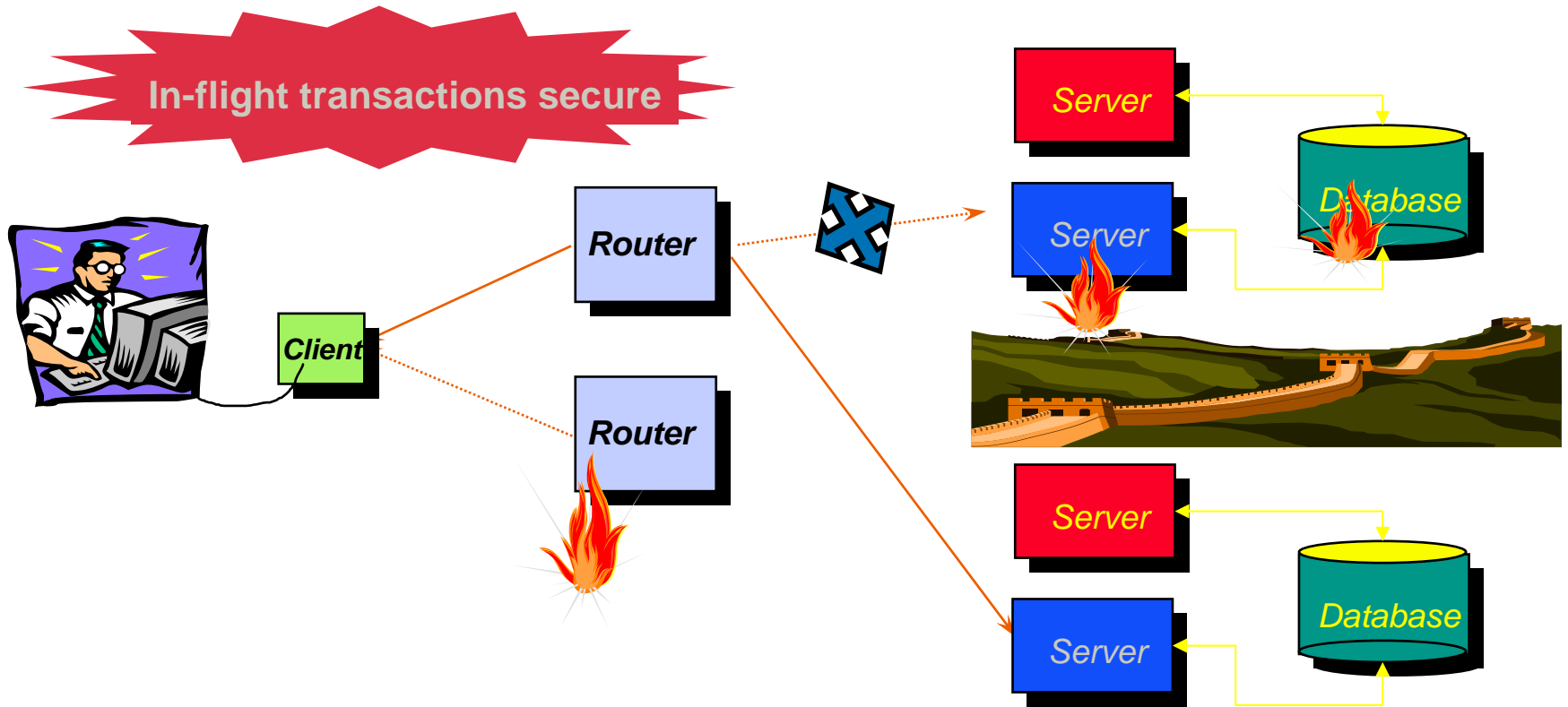


Data is critical



➔ A database can fail

Shadow site – with RTR



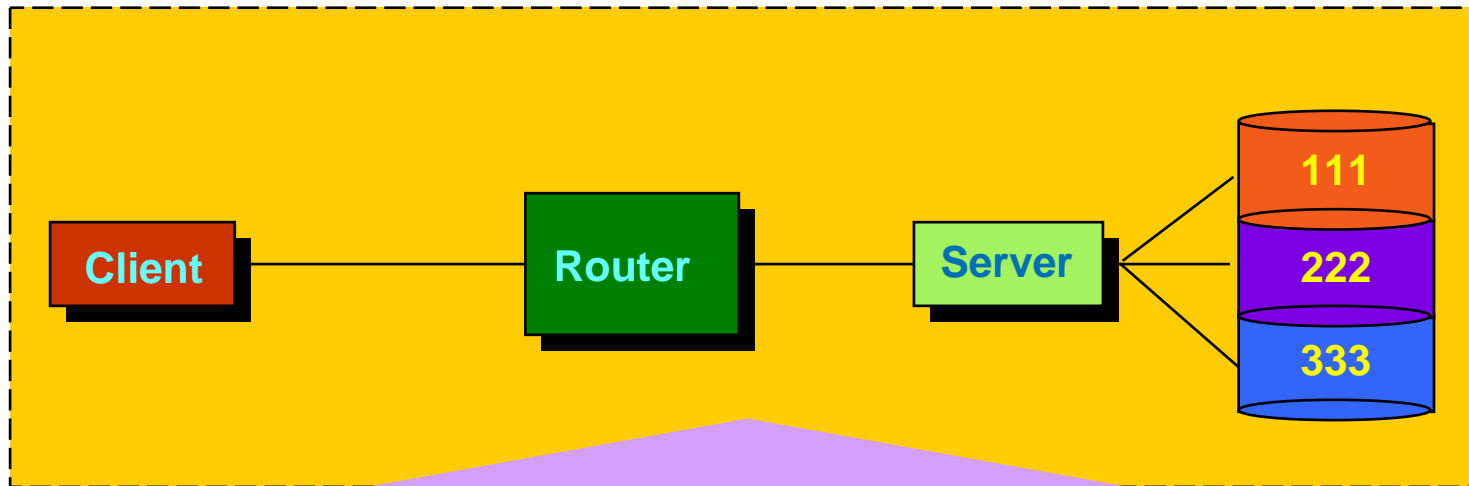
→ Site can fail over to a geographically separate site

Scalability



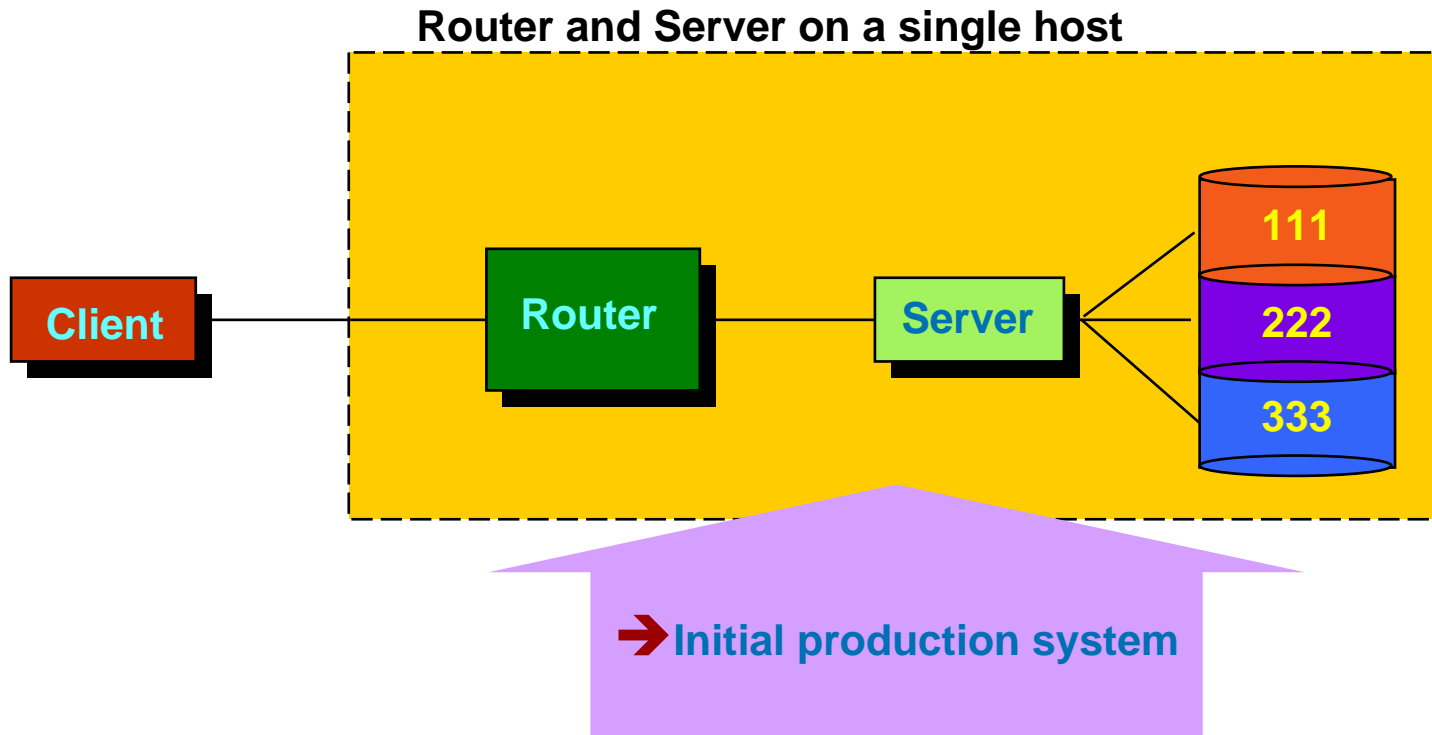
Building scalability with RTR

Single Host Environment



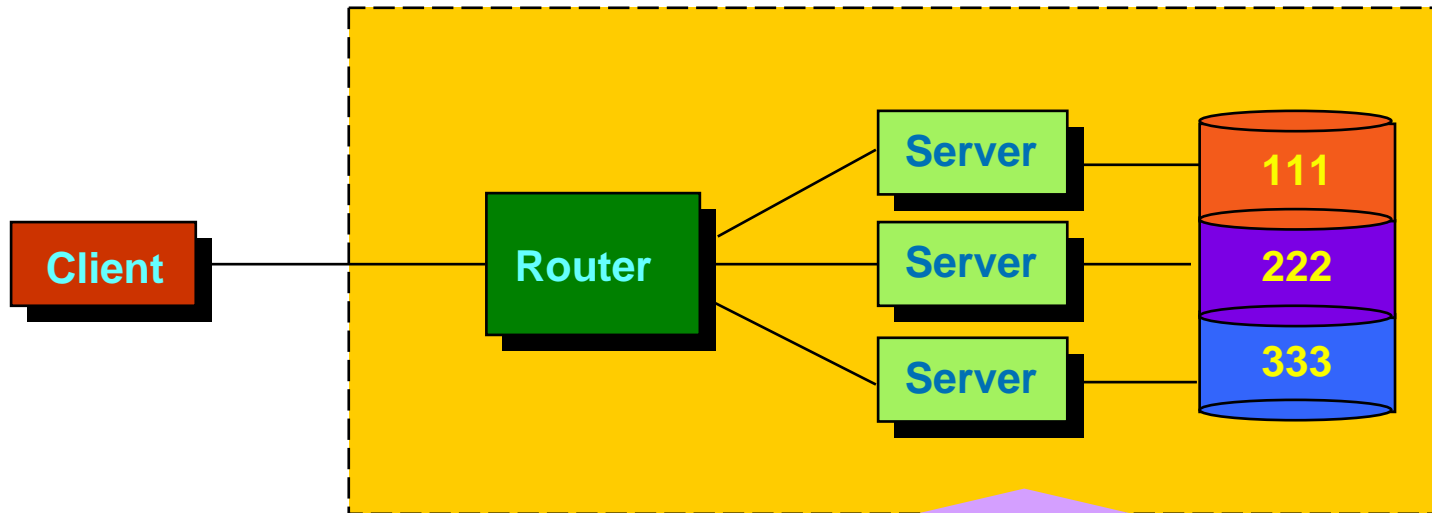
- Development Environment
- Scaled Down Test Configuration
- Basic System – one host with multiple partitions

Scaling up horizontally with RTR



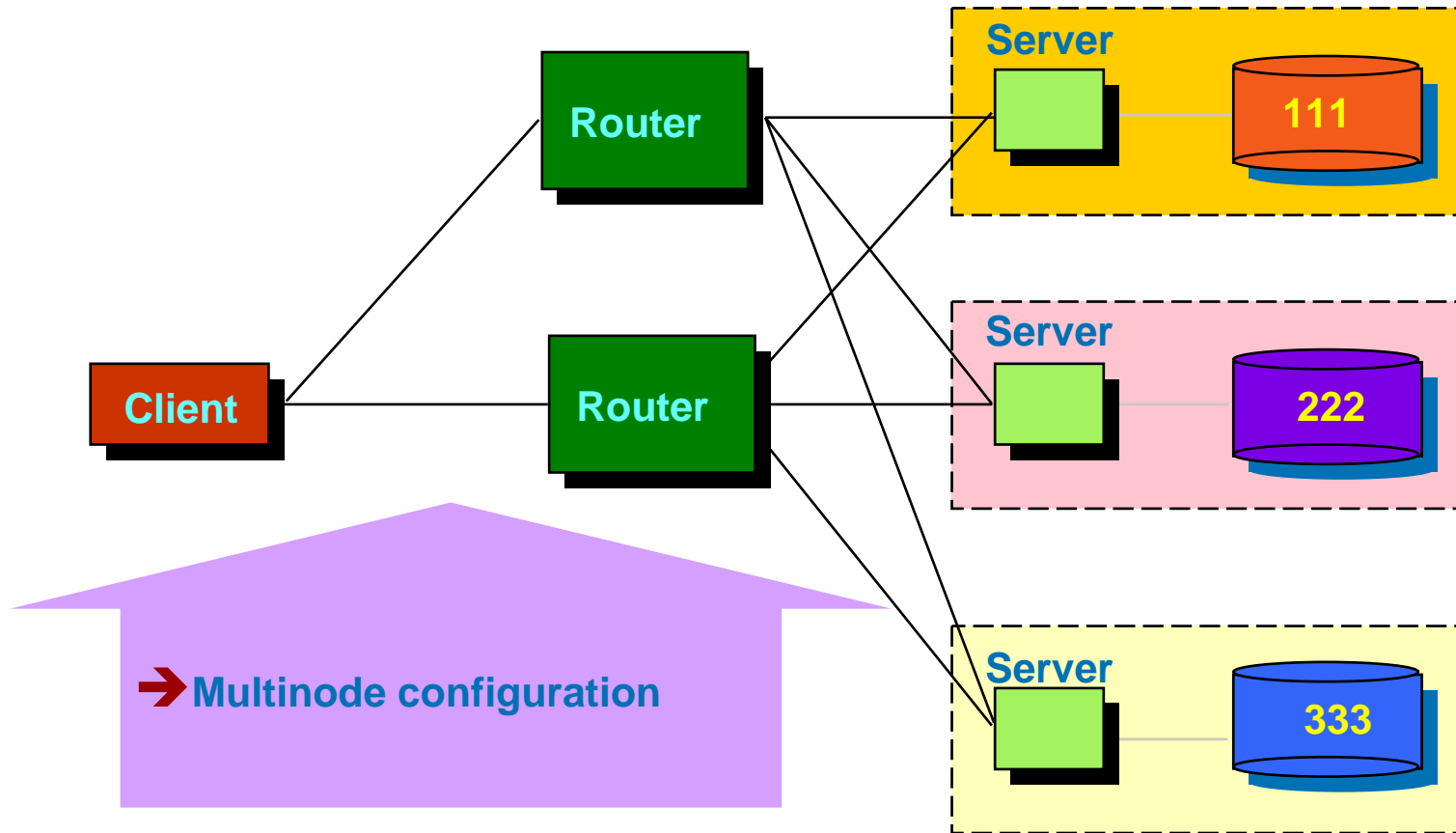
Data partitioning with RTR

Multiple servers on single host



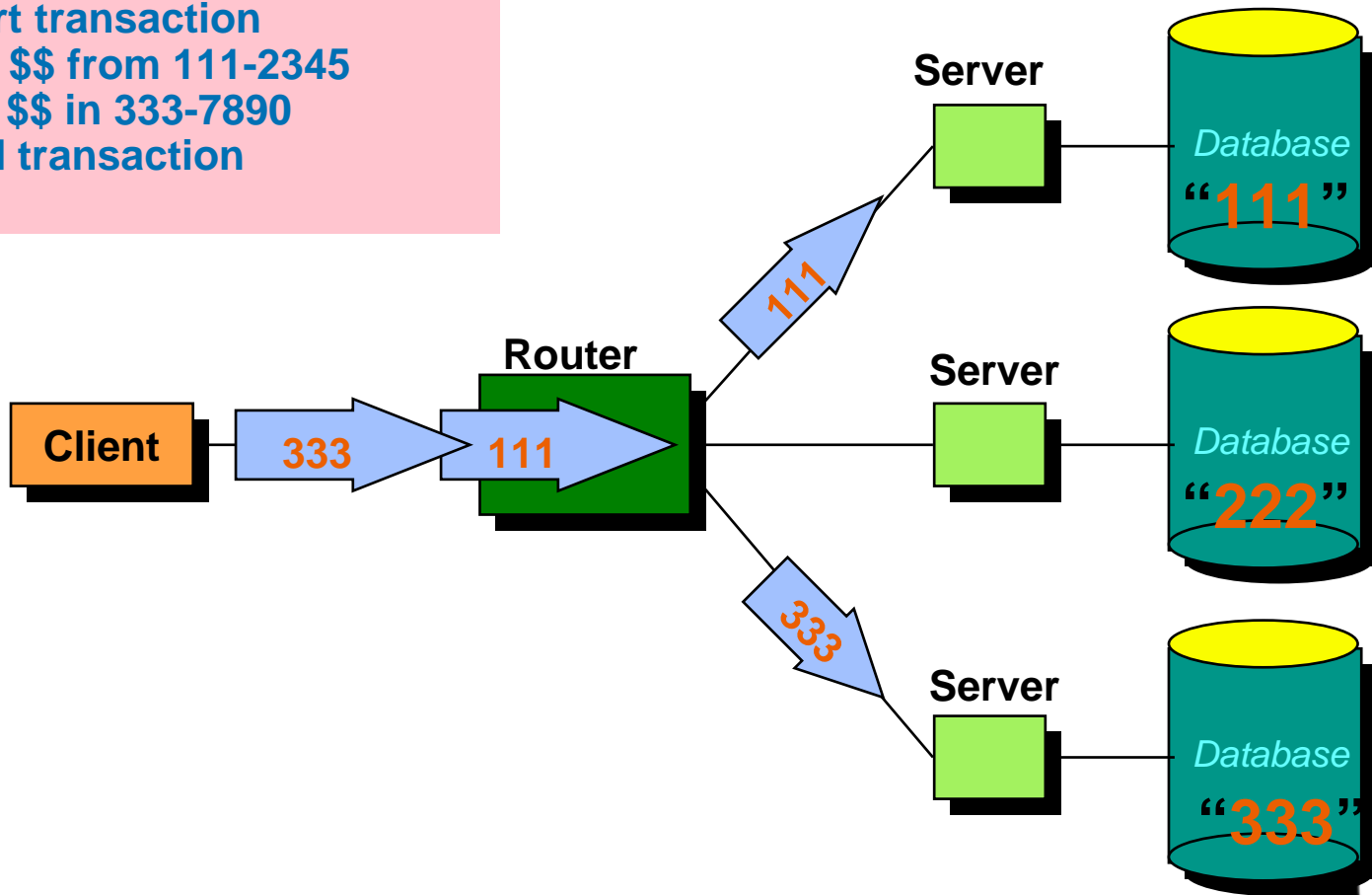
→ Increased performance

More horizontal scalability with RTR



Location transparency with RTR

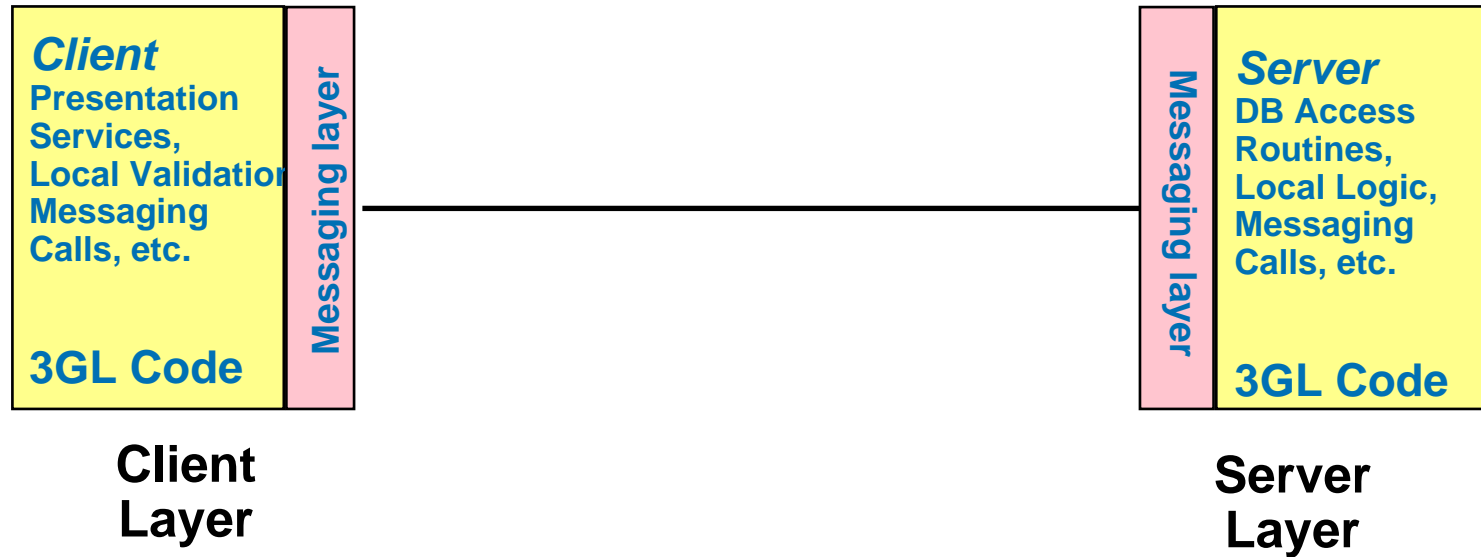
```
{  
  Start transaction  
  Get $$ from 111-2345  
  Put $$ in 333-7890  
  End transaction  
}
```



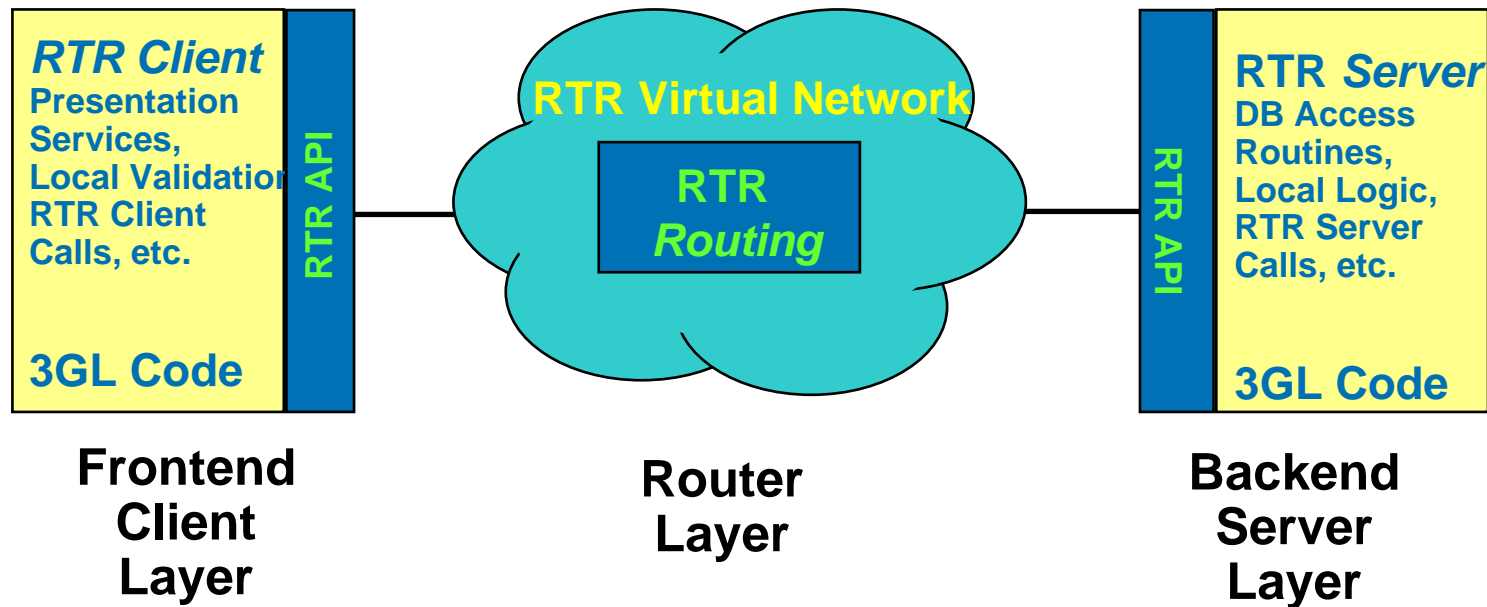
Easy Building & Management



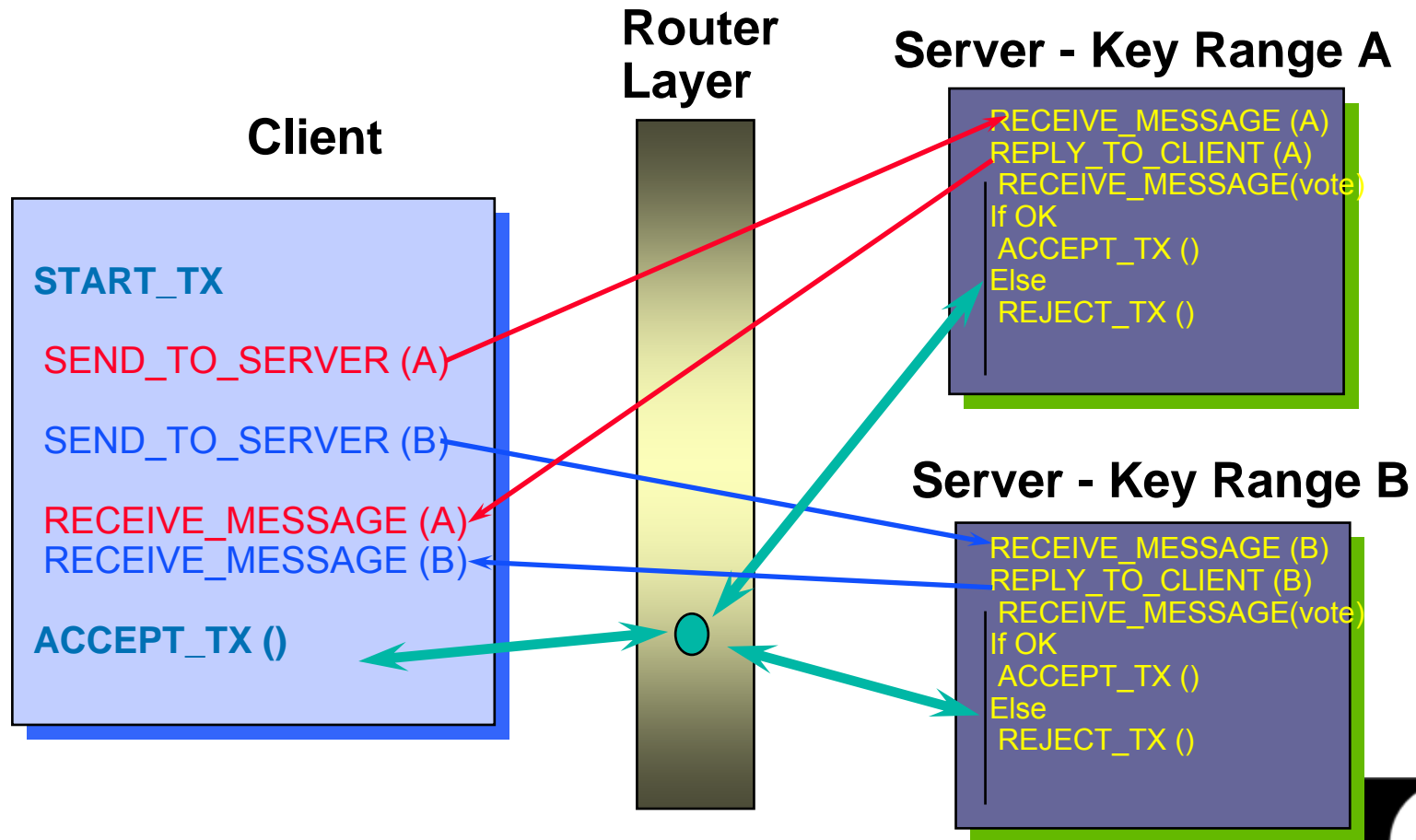
Two-tier client/server architecture without RTR



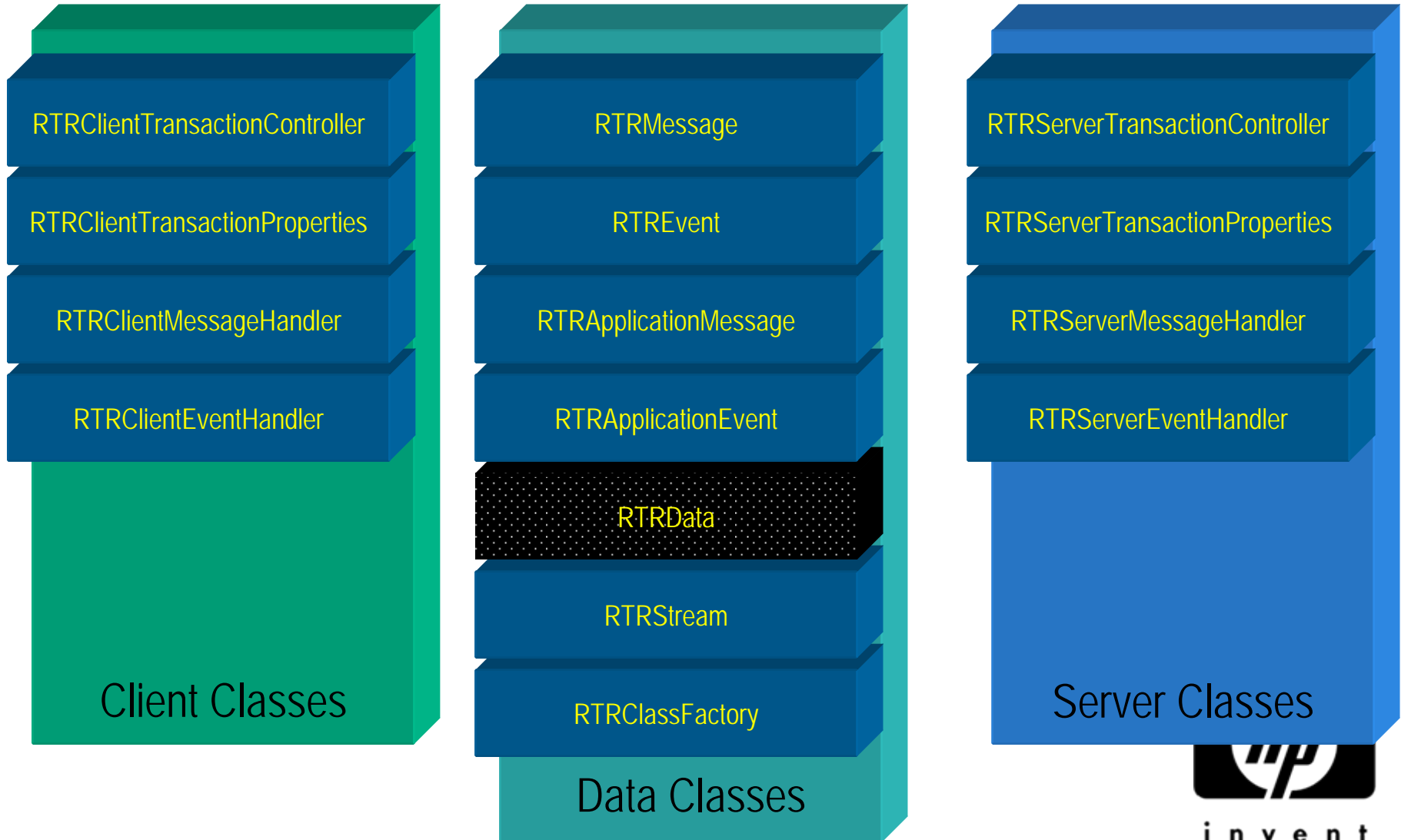
Three-tier client/server architecture with RTR



Simple send-receive C API



Powerful C++ Object Library



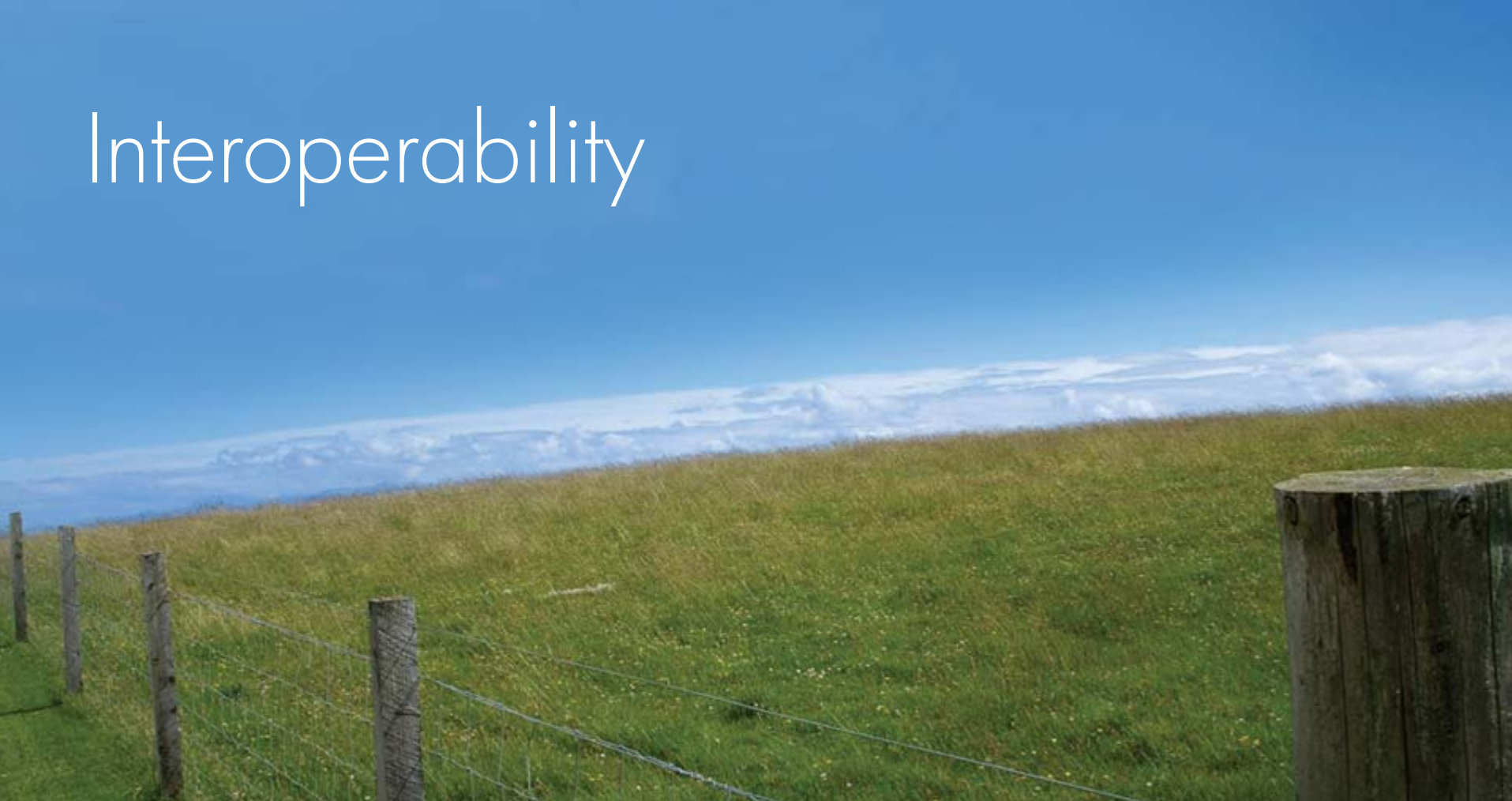
Java Classes in JRTR Toolkit

- Use standard Java streams to send/receive data
- Leverage java.io classes
- Use JTA UserTransaction to define txn boundaries
- Browser-based administration
- JNDI support for JDBC DataSources
- Transparent Connection Pooling

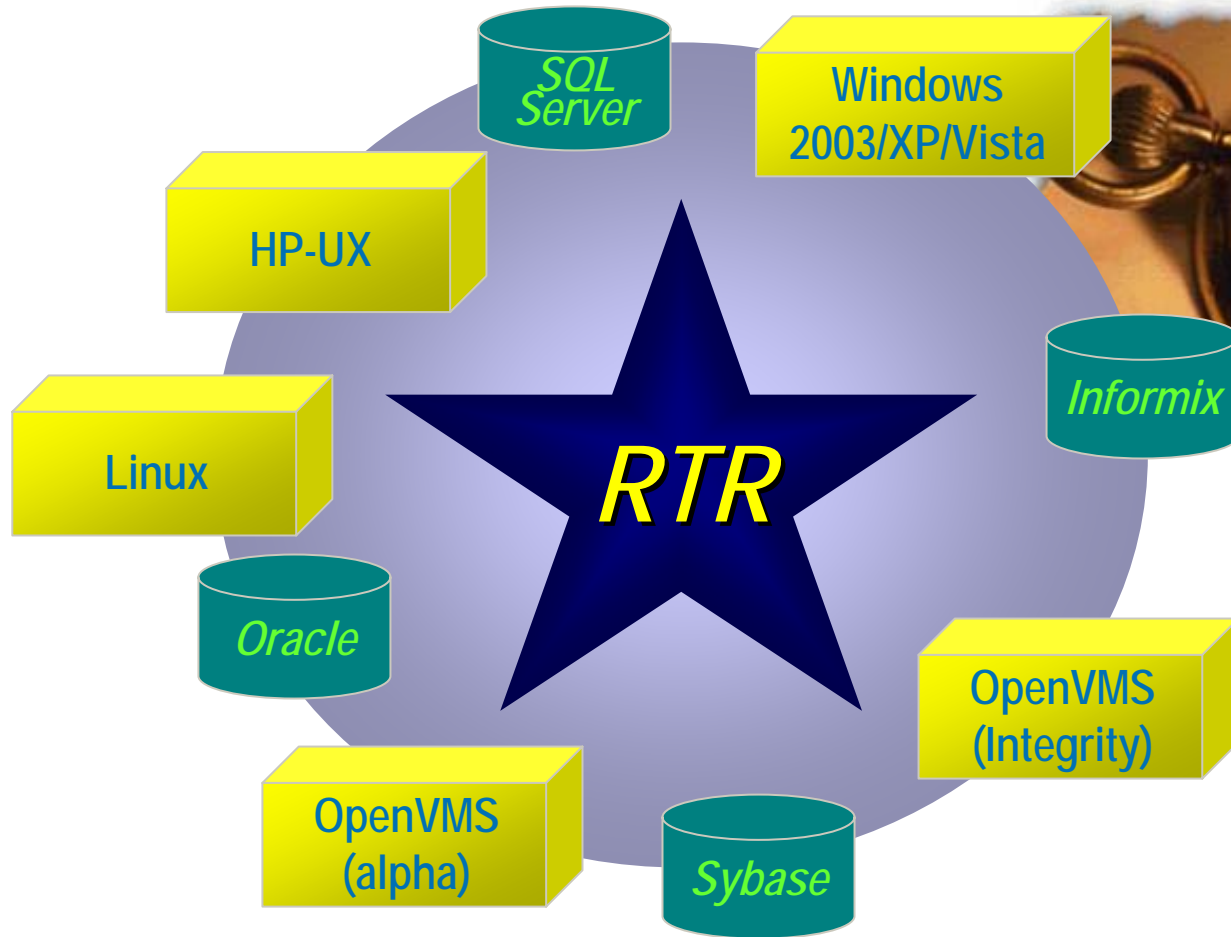
Management Tools

- Standard CLI management interface
- Visual Browser based management capability
- RTR Explorer for self diagnosis and rapid problem location
- Performance monitoring using Windows Performance Monitor

Interoperability



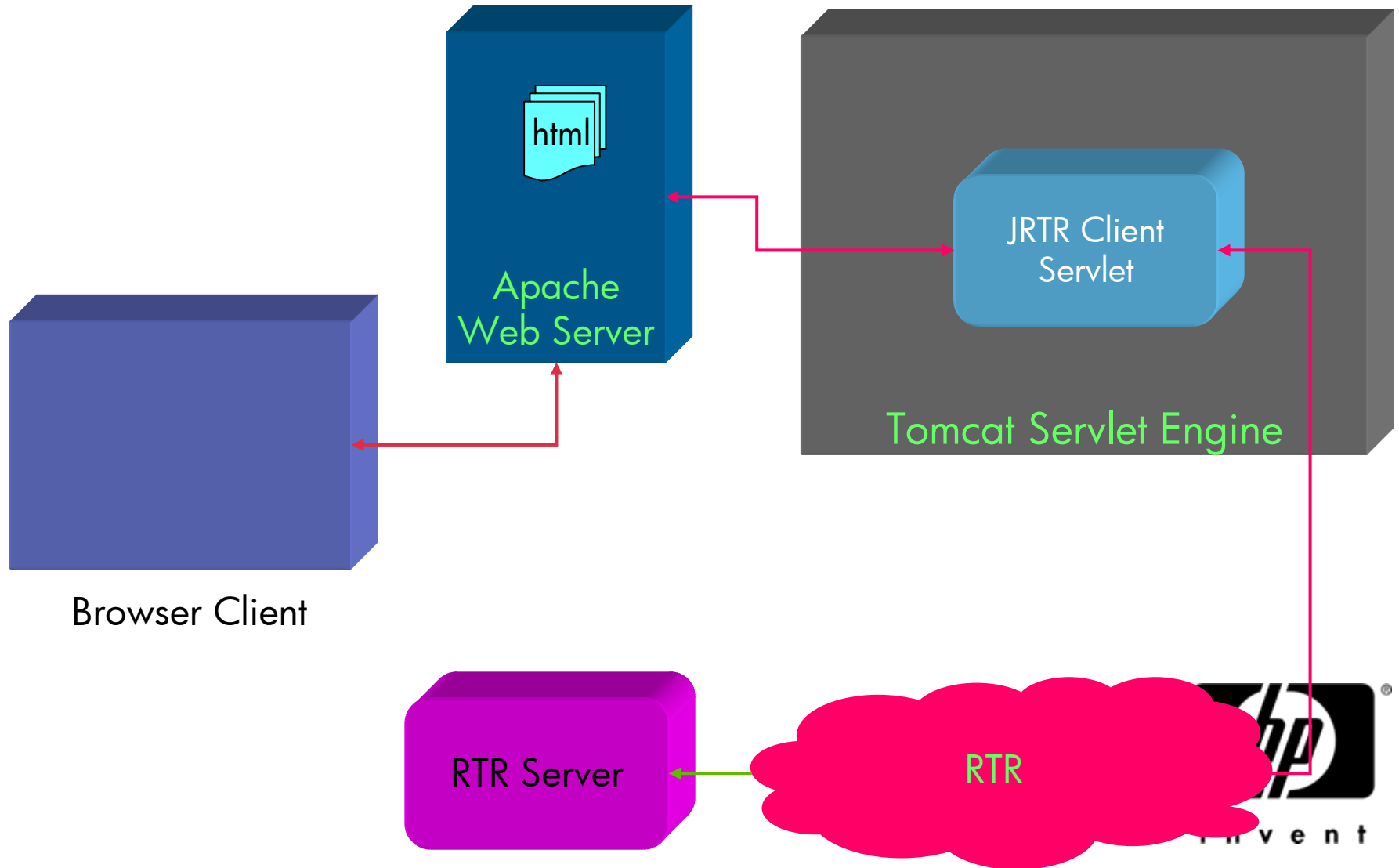
Interoperability



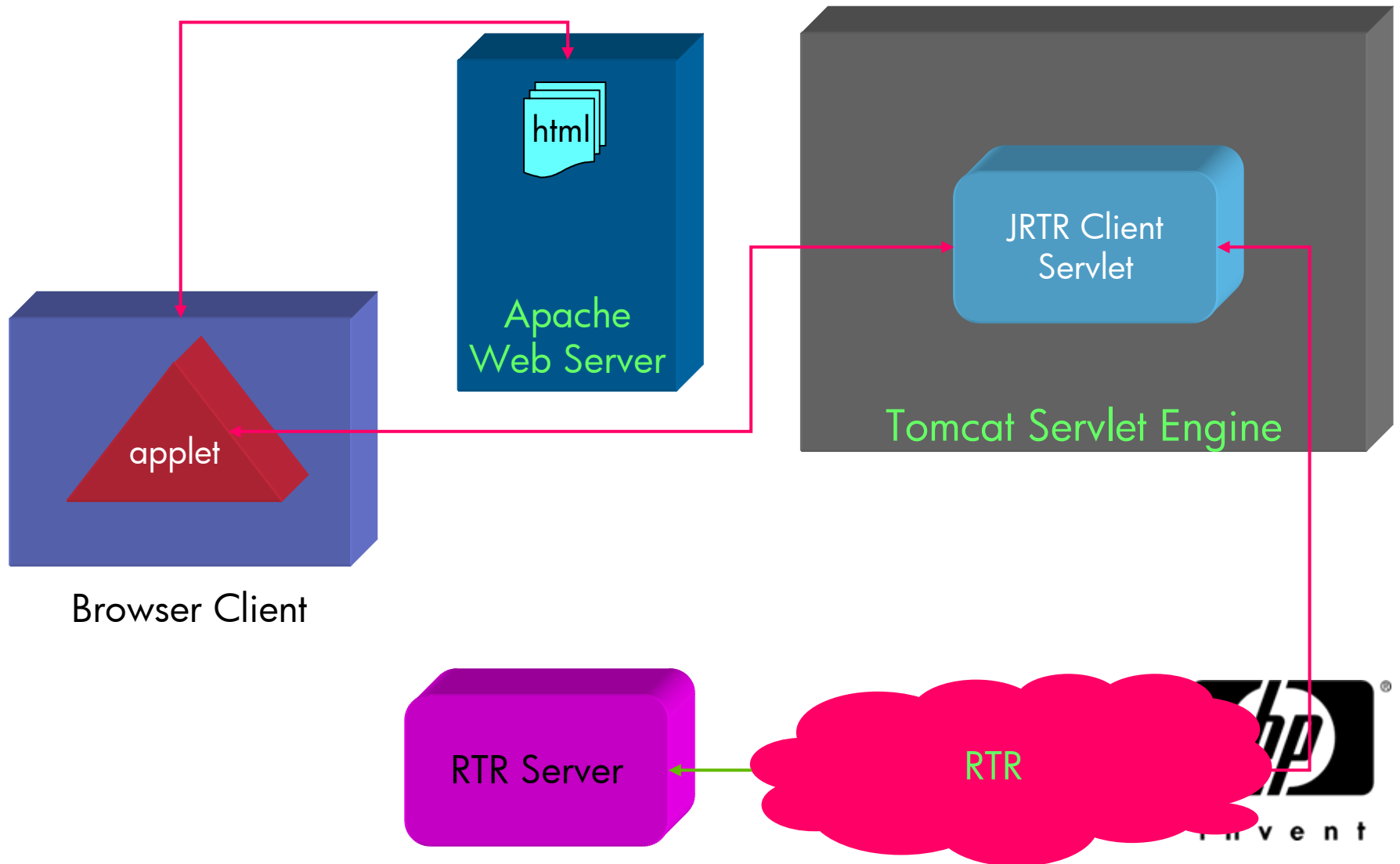
Web Applications



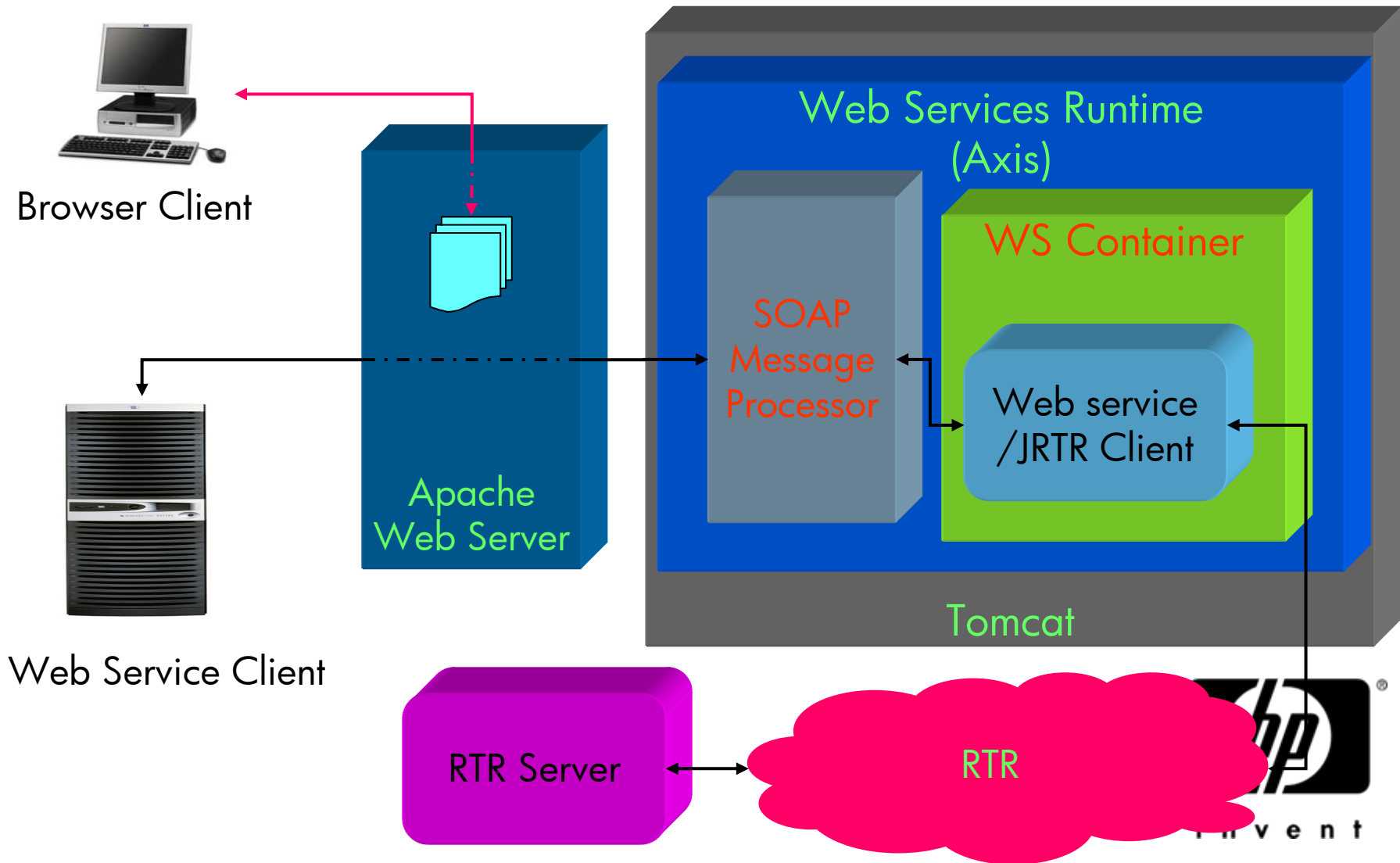
JRTR With Java Servlets



JRTR With Applets & Servlets



Web Services with JRTR/RTR



To know more

- Product Management
 - Rick McLaughlin
Product Manager

+1 603 884 0992

http://www.hp.com/products1/rtr/fb_rtrbusmgr.html

- RTR Web Site: www.hp.com/go/rtr

