Large-Scale System Integration with DDS for SCADA, C2, and Finance

Rick Warren, Principal Engineer rick.warren@rti.com

What do I mean by “large system”?

- **Systems of systems**
  - Modular, hierarchical design
  - Legacy components, subsystems
  - Multiple technologies
  - Global scale
- **Decoupled subsystem lifecycles**
  - Independent development
  - Independent deployment and use
  - Independent management
  - Independent revision and retirement
- **Multiple communities of interest**
  - Different data interest, entitlements
  - Non-uniform levels of trust

What matters to integrators of large systems?

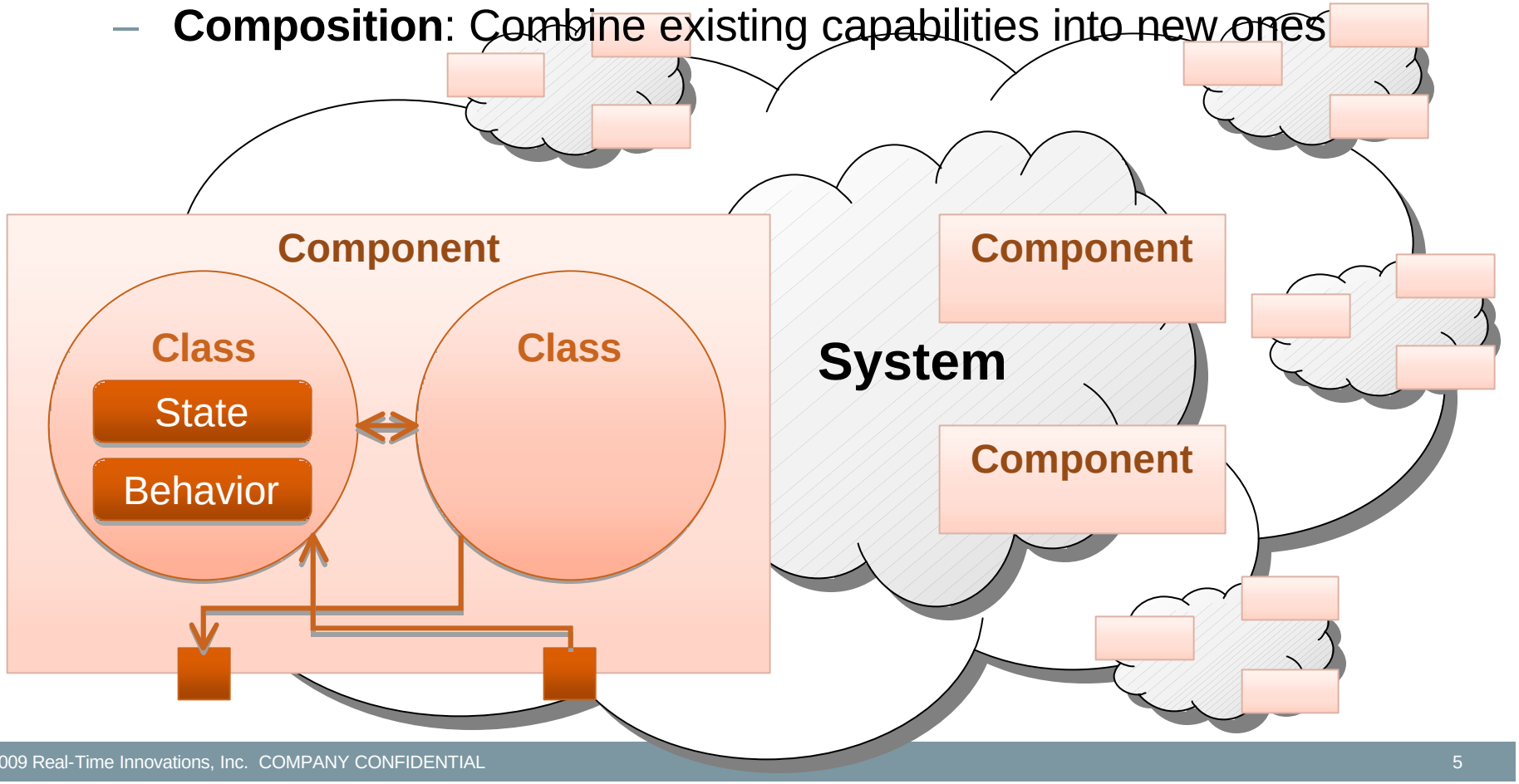
- **Governance**
  - **What** information will be exchanged?
  - **Under what conditions** will it be exchanged?
  - **Who** is allowed to exchange this information?
  - **If these SLAs are violated**, can the exchange be prevented? Can I be notified?
  - **In the past**, what has occurred wrt these SLAs?
- **Isolation**
  - When I connect A and B, ensure they don't break (each other)
  - When I disconnect A, ensure B doesn't break
  - When I connect C, don't change A or B
- **Scalability** *(like in any system,*
- **Fault Tolerance** *but stakes are higher)*

Part 1: Architecture

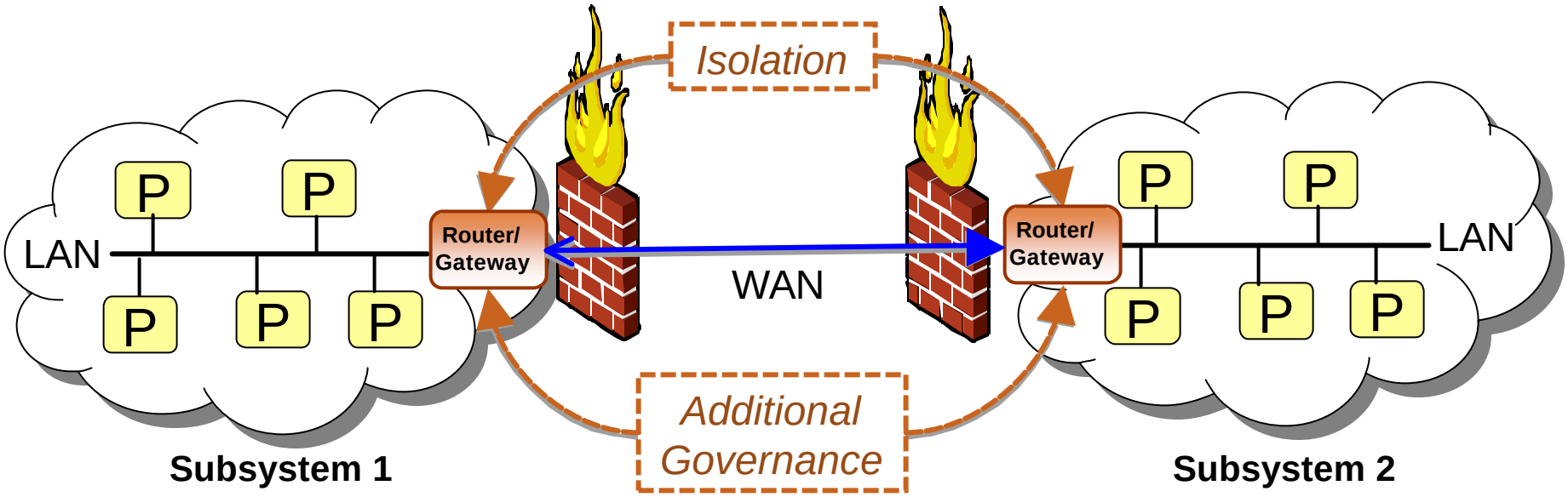
(we'll get to technology later)

A Modest Proposal

- ***Fundamental design principles scale***
  - **Abstraction**: Provide interface based on relevant concepts
  - **Encapsulation**: Hide internal implementation, communication
  - **Composition**: Combine existing capabilities into new ones

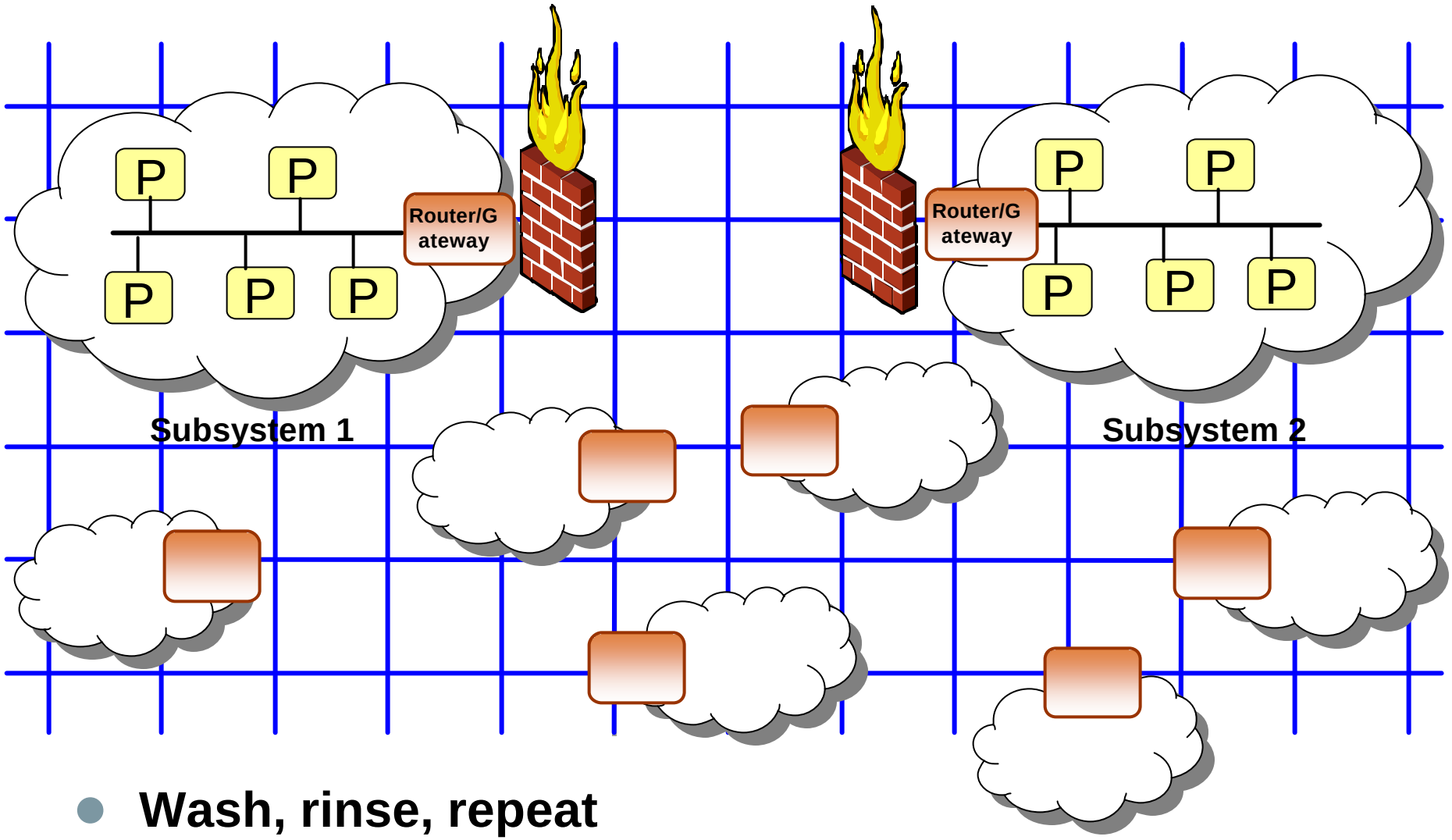


Schematic of a Composed System



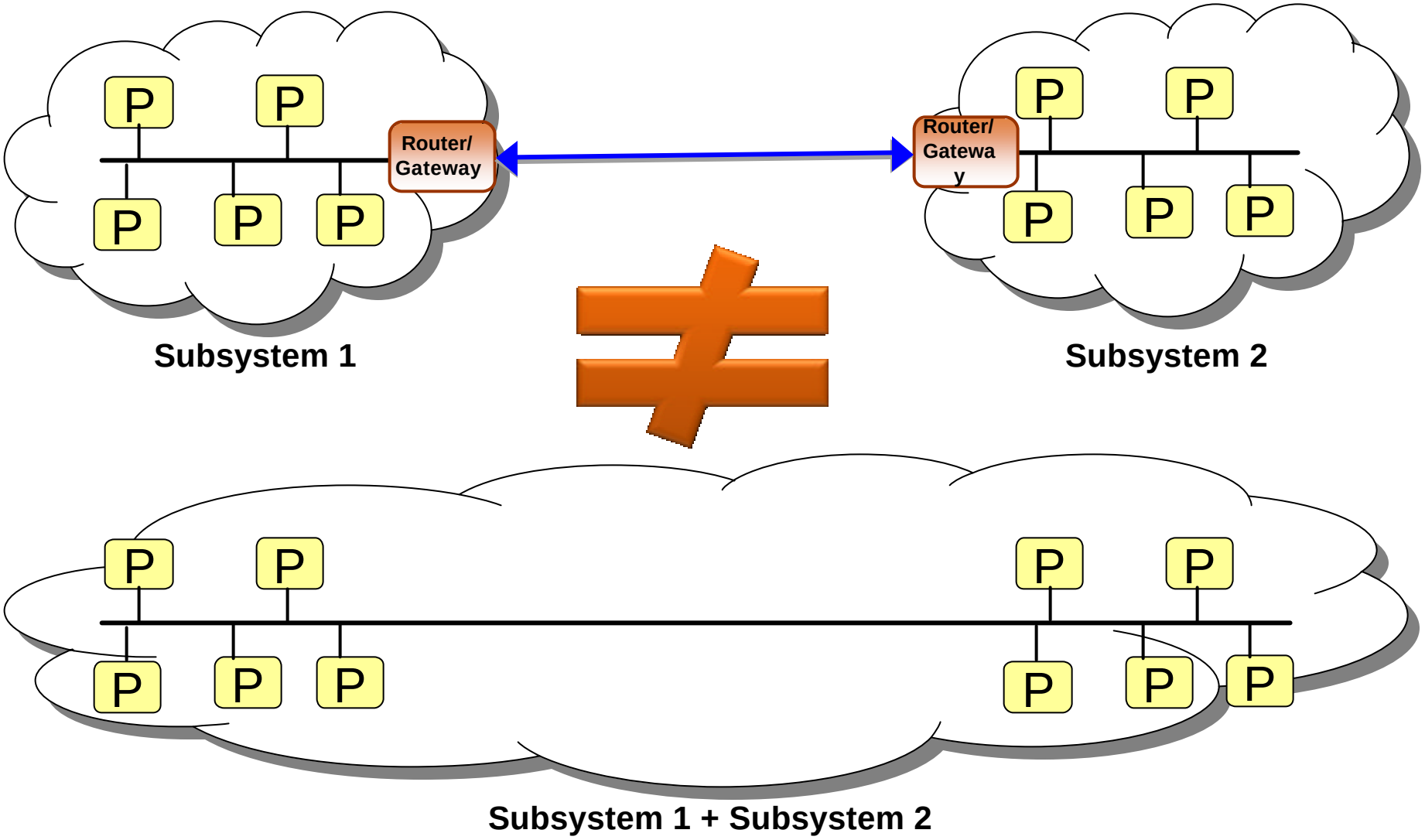
- Subsystems may have different network environments
- Integration may have different network environment than subsystems themselves
- Data may need to be transformed/cleansed as it moves among subsystems
- *Routing/gateway services will adapt data types / formats / protocols*

Schematic of a Composed System

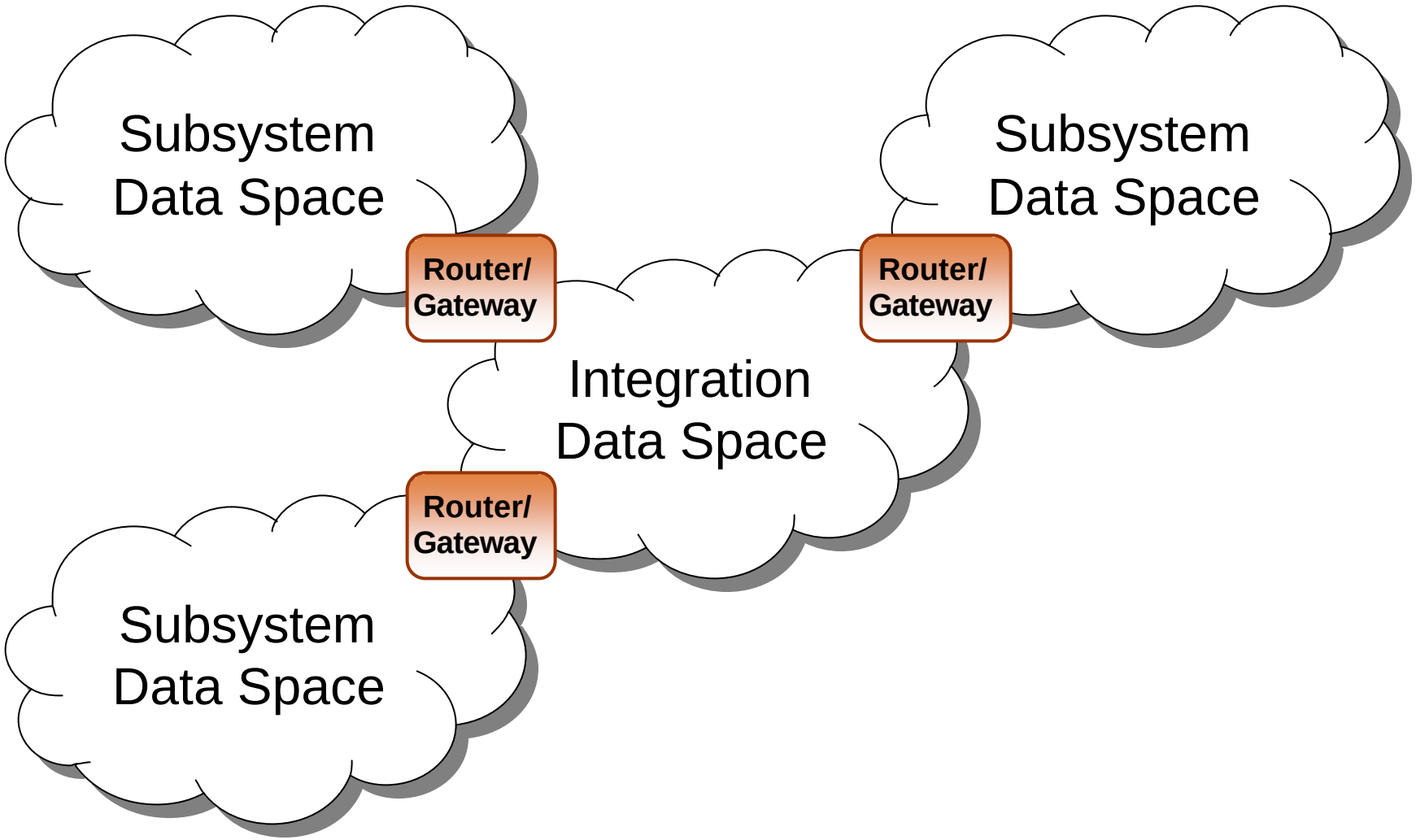


- **Wash, rinse, repeat**

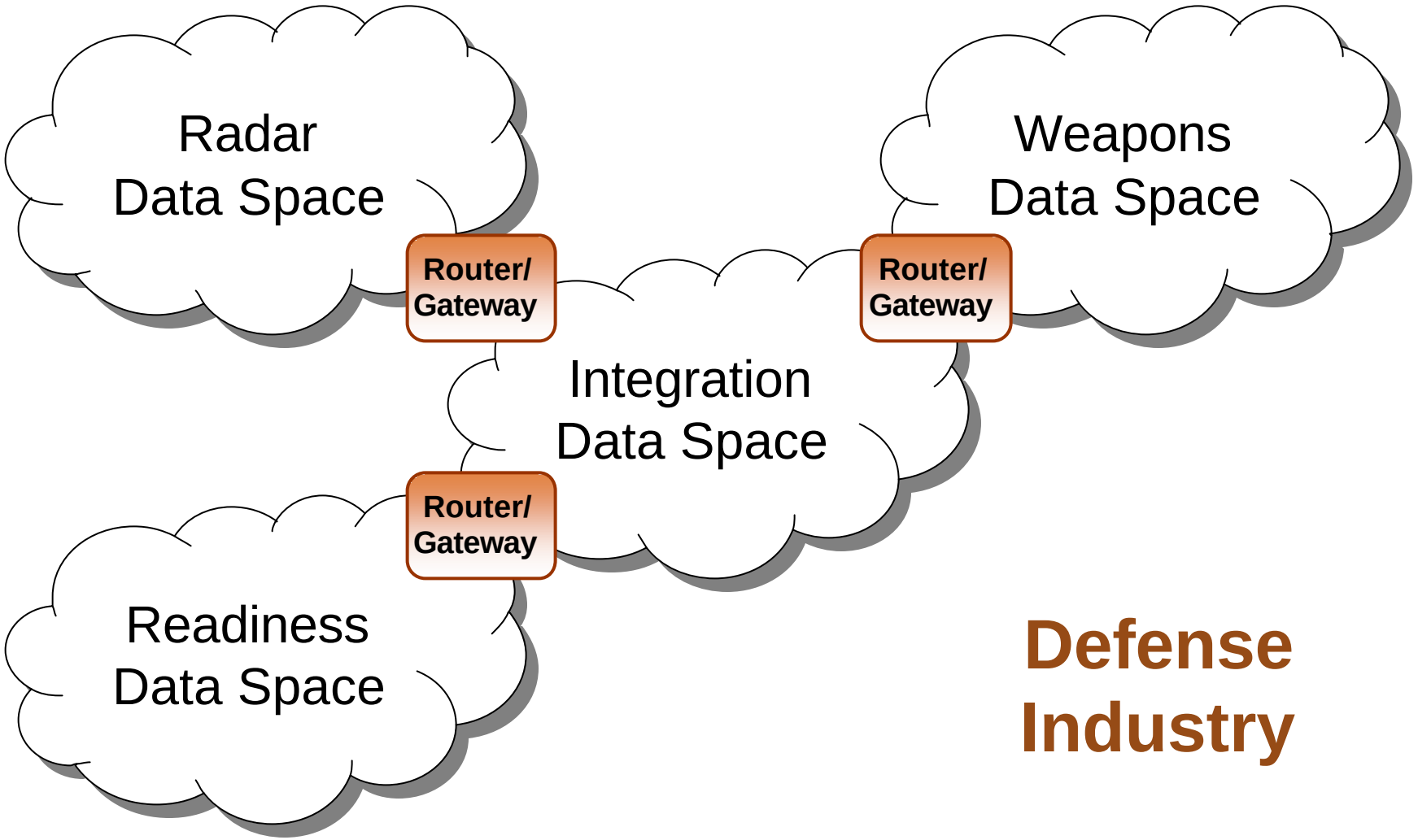
Apples and Oranges



Nothing New Under the Sun

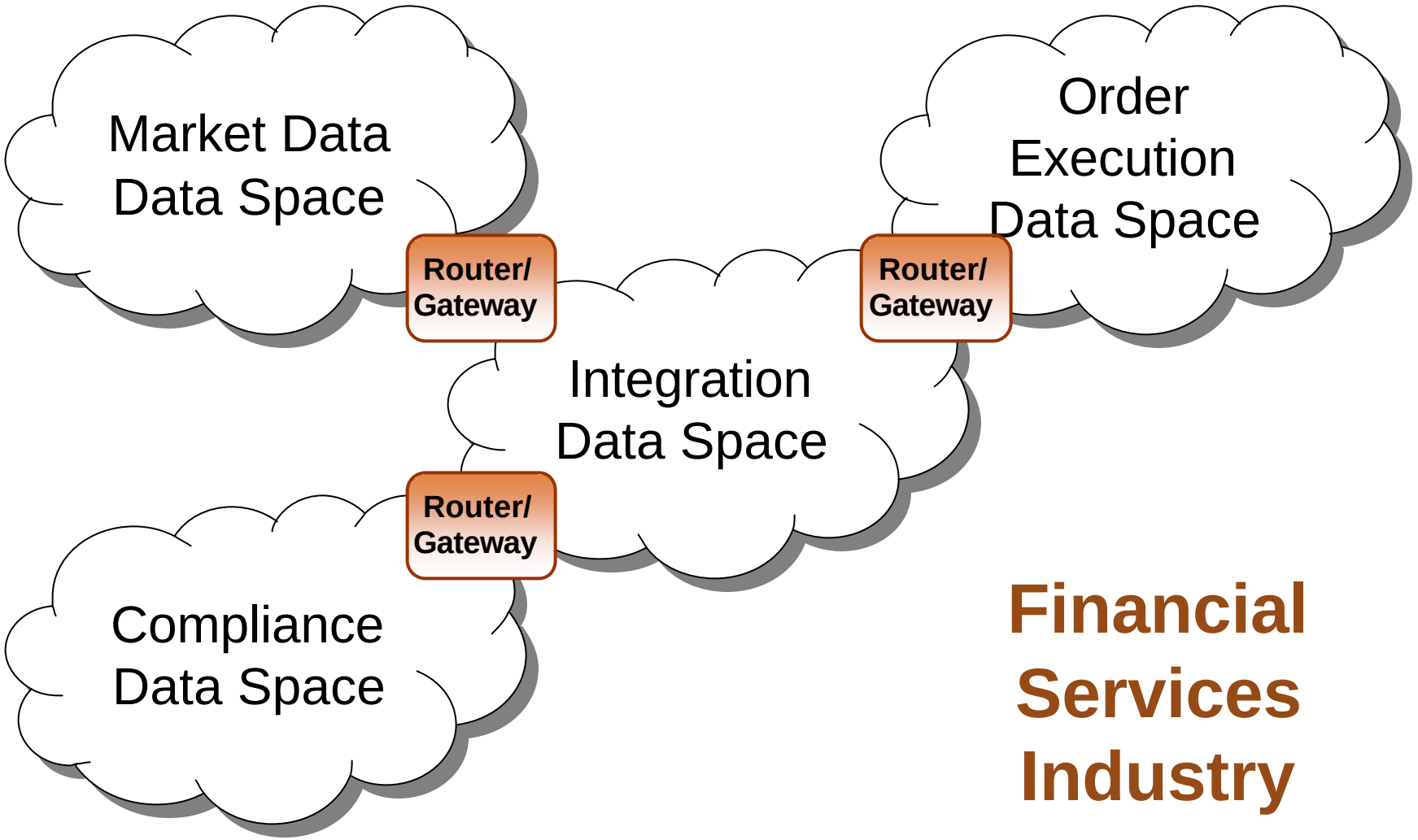


Nothing New Under the Sun



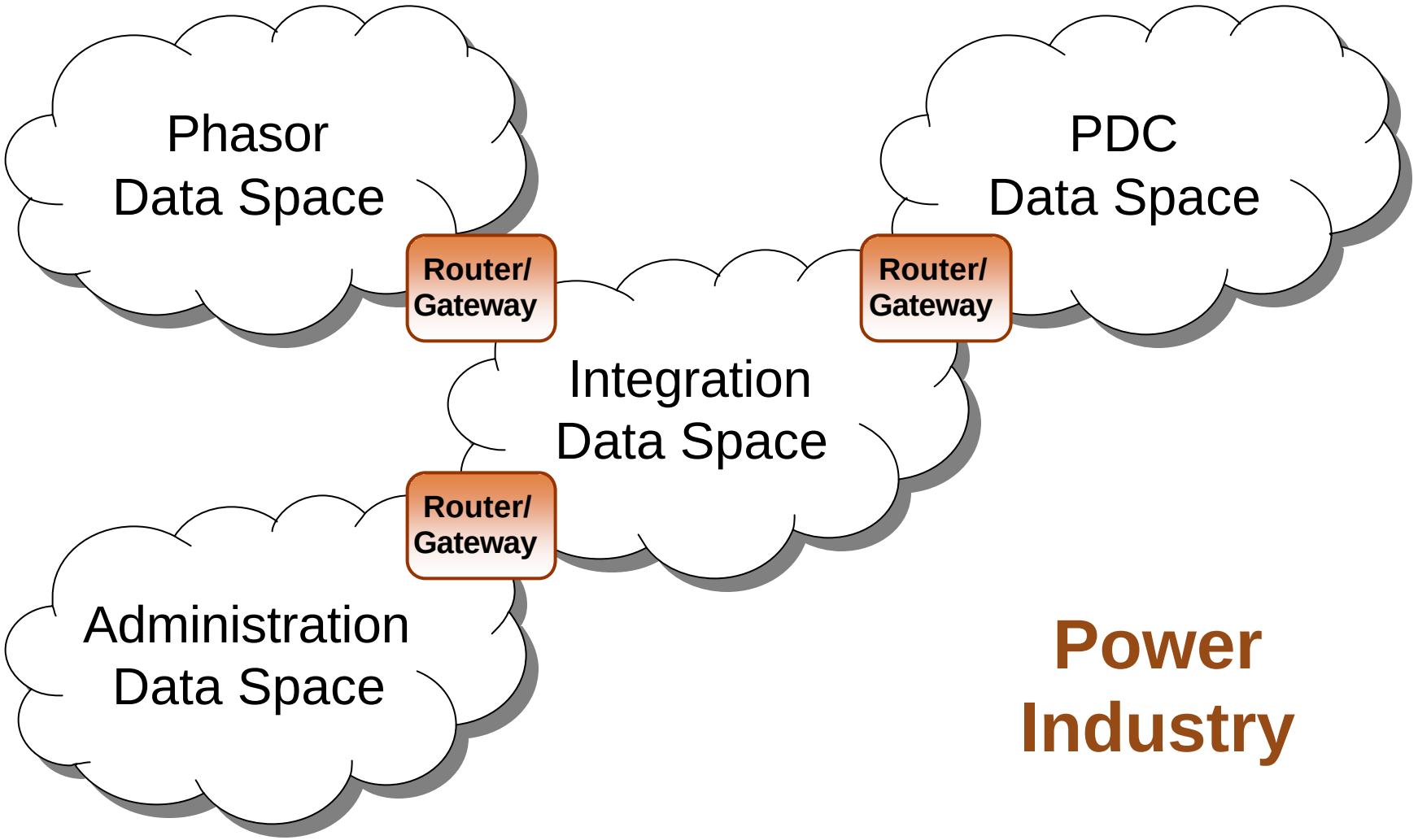
Defense Industry

Nothing New Under the Sun



Financial Services Industry

Nothing New Under the Sun



Power Industry

Architecture Recap

- Solve big problems like you solve small ones
  - Break them into pieces
  - Define interfaces between pieces
  - Implement each piece
  - Integrate along the interfaces
- Translation
  - Define subsystems
  - Implement them how you like
    - May have different requirements, therefore technology
  - Router/gateway defines/enforces interfaces

Remaining Problems to Address

- **Fault tolerance**: Router can't be single point of failure
  3 answers:
  - Persistent data survives system failures
  - Segmented/load-balanced configuration limits scope of failures
  - Redundancy allows continued service during failure

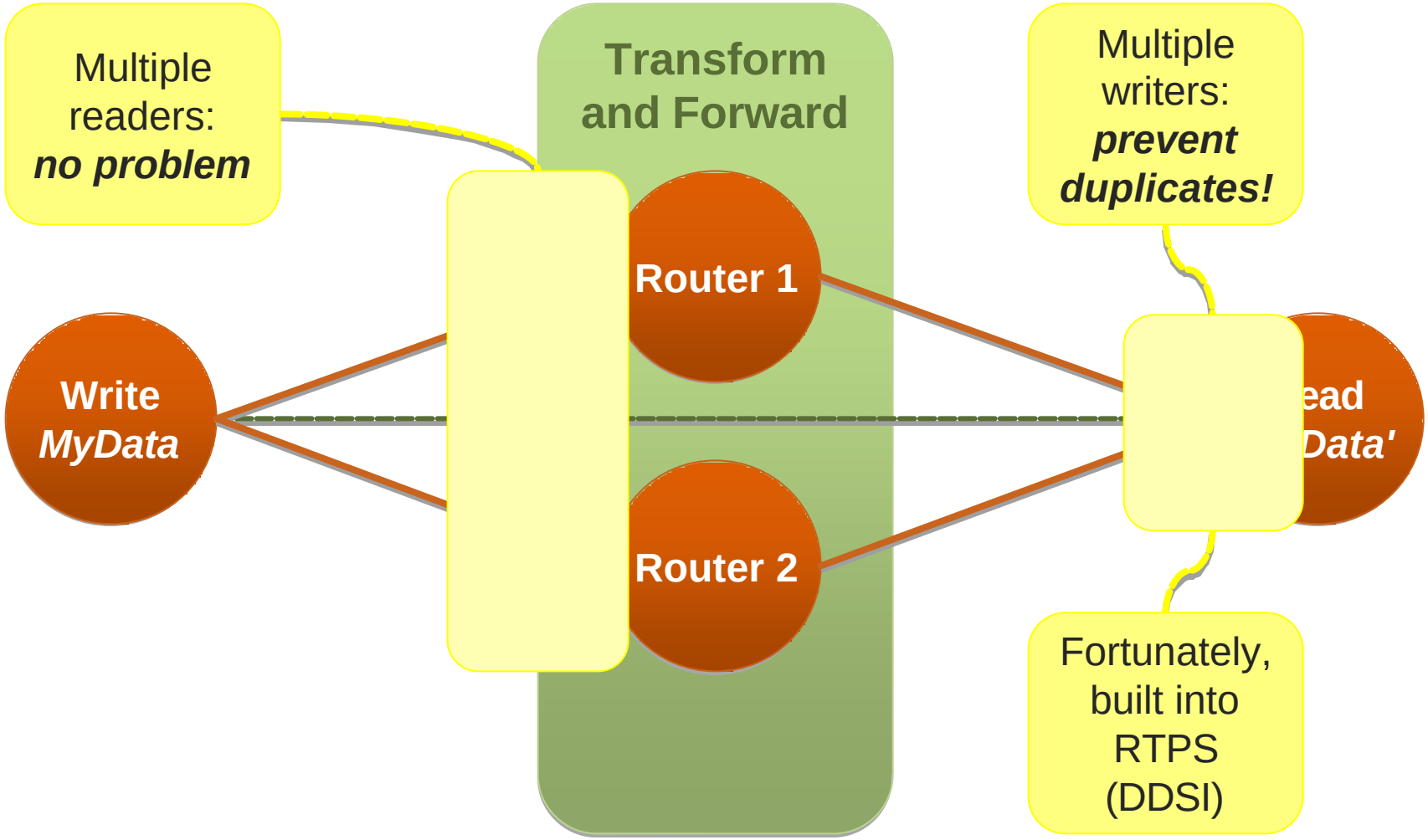
- **Scalability**: How big can a subsystem be?
  1. How big a subsystem can you understand, test, and debug?
  2. How well does infrastructure scale?

Part 2: DDS

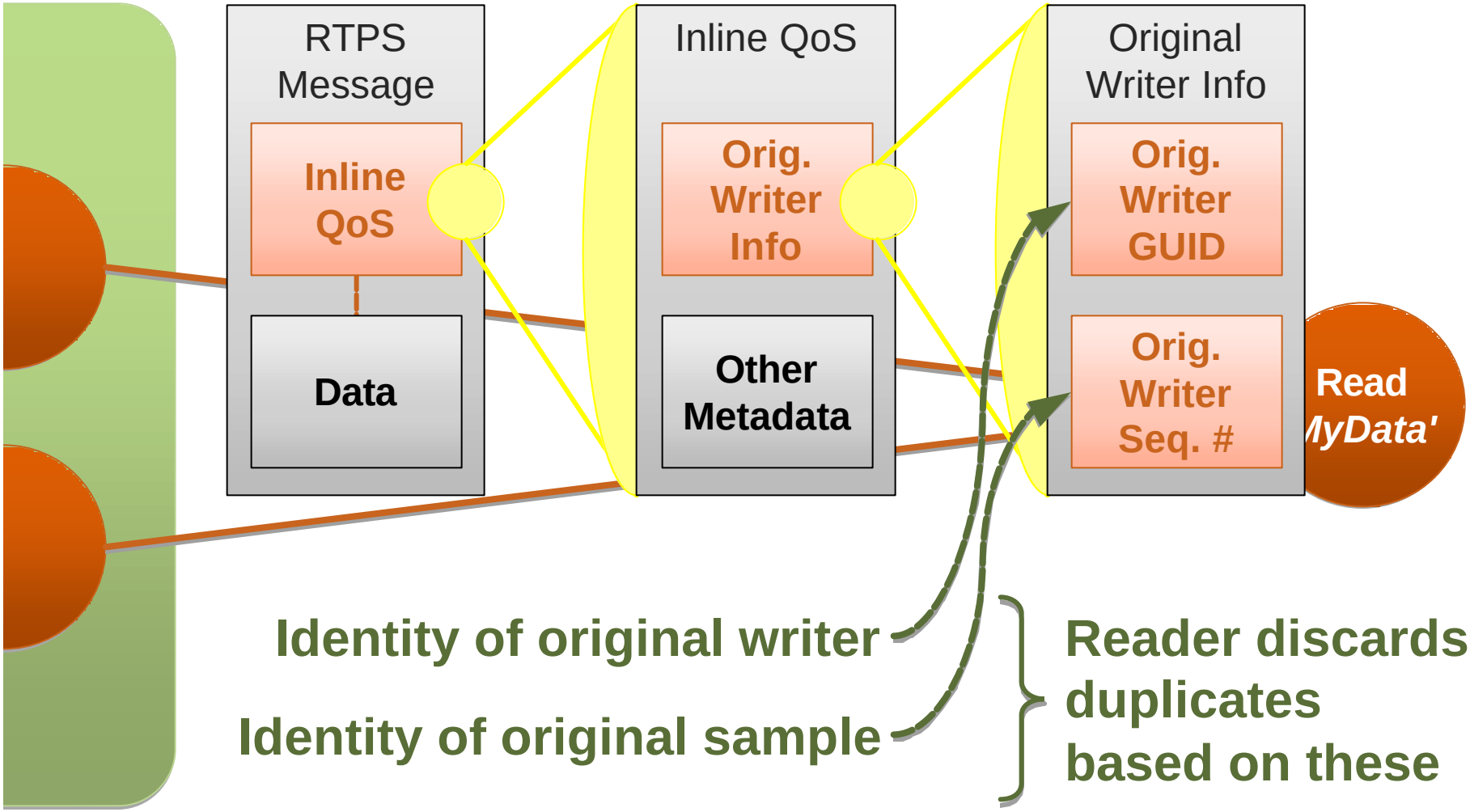
How Does DDS Stack Up?

- Governance
  - Structural SLA: data type — Built in
  - Behavioral SLA: delivery QoS — Built in
  - Security — Products available; future standards
  - Problem detection/prevention/notification — Built in
  - System monitoring and recording — Products available
- Isolation
  - Prevent data “leaks” and side effects — Built in: domains, partitions
  - Allow dynamic (dis-|re-)connection — Built in
  - Support independent integration and evolution — Built in, esp. w/ DDS-Xtypes
- Fault tolerance
  - Data persistence — Built in
  - Segmented/load-balanced routing — Built in if router uses DDS
  - Redundant routing — Built in if router uses DDS
- Scalability
  - WAN connectivity — Products available; future standards
  - Peer-to-peer communication — Highly scalable
  - Brokered/managed communication — Protocol support if necessary

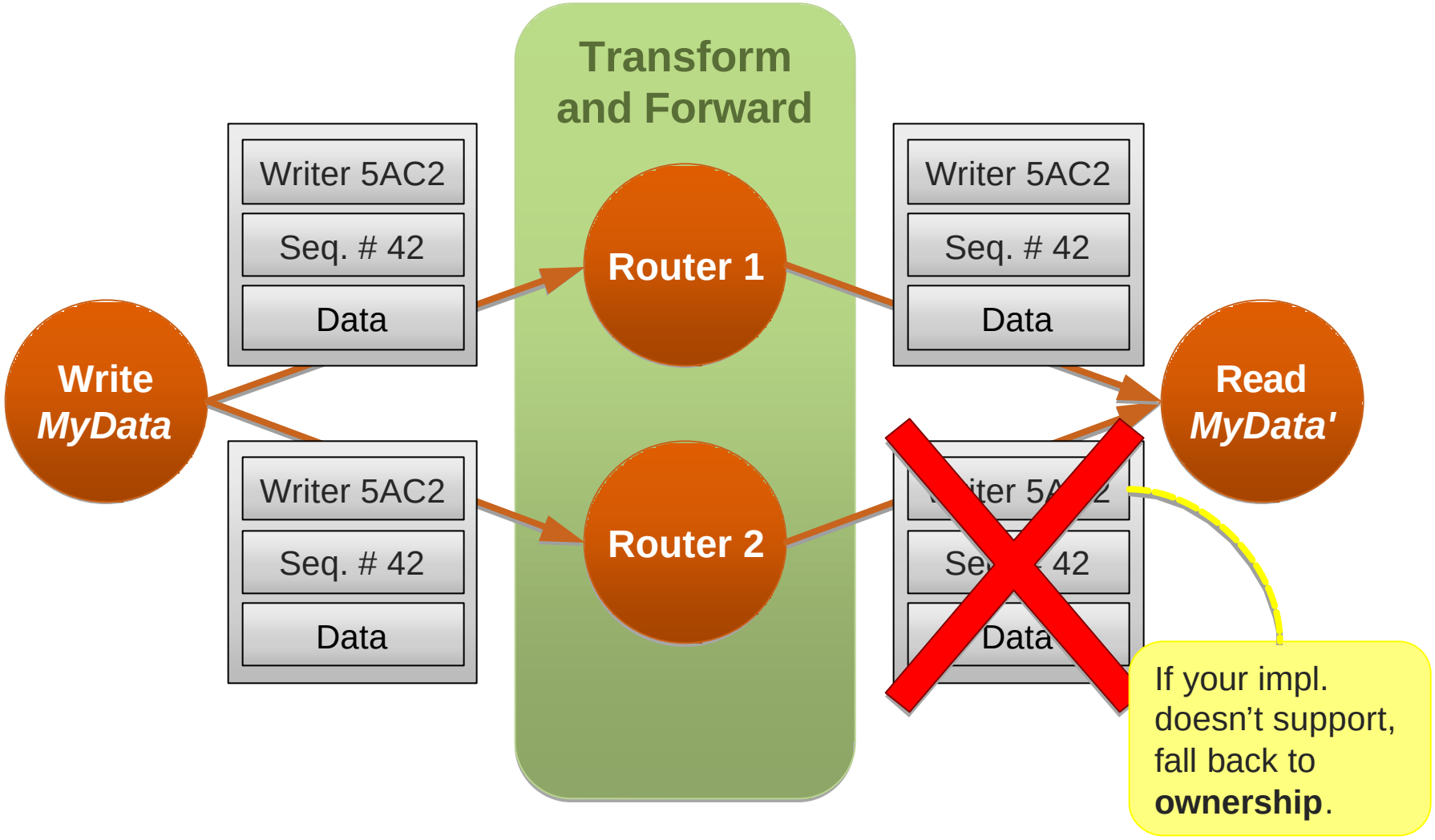
Redundant Data Routing



Redundant Data Routing



Redundant Data Routing

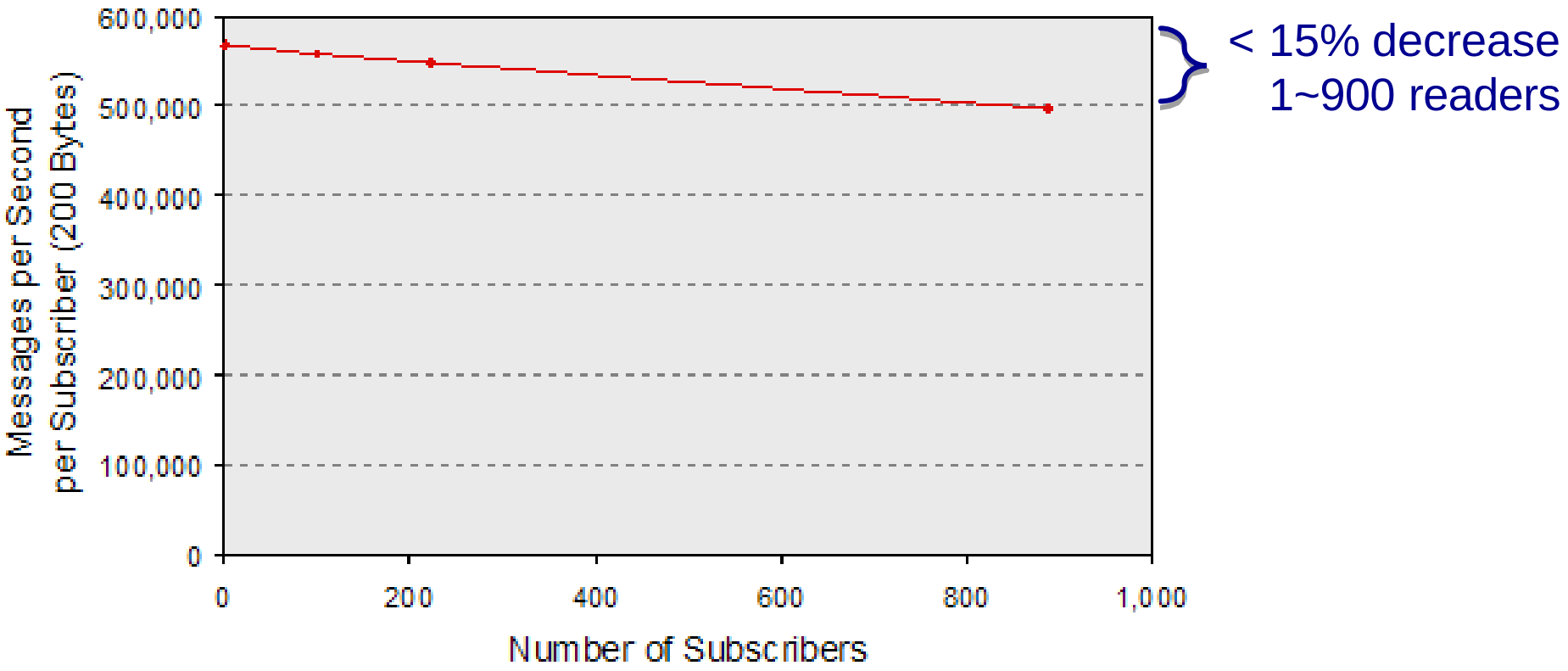


DDS Scalability: Two Aspects

- **Application data**
  - How does performance fall off as # participants, writers, readers increase?
  - Any single writer/reader pair can saturate gig-E: aggregate throughput is not main issue
  - *Interesting issue*: Fan-out – number of readers per writer
- **Discovery**
  - How many DDS applications can discover one another?
  - *Interesting issue*: How many applications *need* to discover one another?
- *Hard limits, if any, very dependent on DDS implementation, machine configuration, network, etc.*

DDS Scalability: Application Data

- RTPS reliable multicast scales *at least*:
  - …to hundreds of readers per writer
  - …with very little degradation in throughput.
  - Larger testing facilities welcomed. ☺



DDS Scalability: P2P Discovery Scenarios

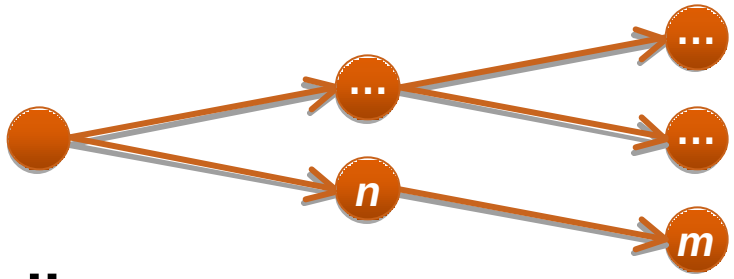
- **Single domain, symmetric discovery**
  - Everyone discovers everyone else
  - Easiest to configure; most challenging wrt scalability
  - *RTI test results*: **1,800 participants**; **3.2M endpoint matches**
- **Single domain, asymmetric discovery**
  - Each participant only knows of certain others in its domain
  - Good for partitioning domains in which not everyone talks

- **Multiple domains**
  - Greatest separation for subsystems that rarely exchange data
    - RTPS maps to different IP ports

DDS Scalability: Stronger Measures

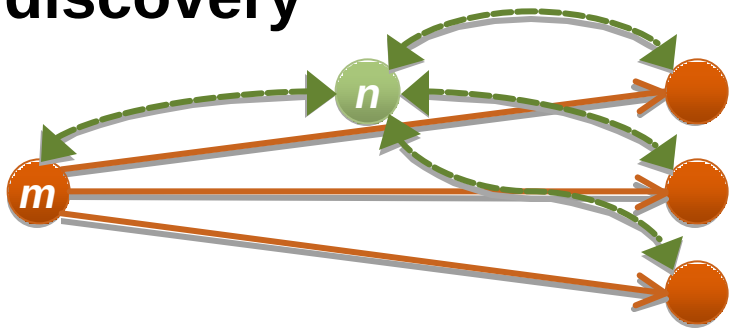
- P2P RTPS allows participant to talk to at least 1-2K others (*based on current experimental data*)
  - My system is properly decomposed. Discovery is optimized. *But I still need to scale bigger.*

1. **Brokered application data**



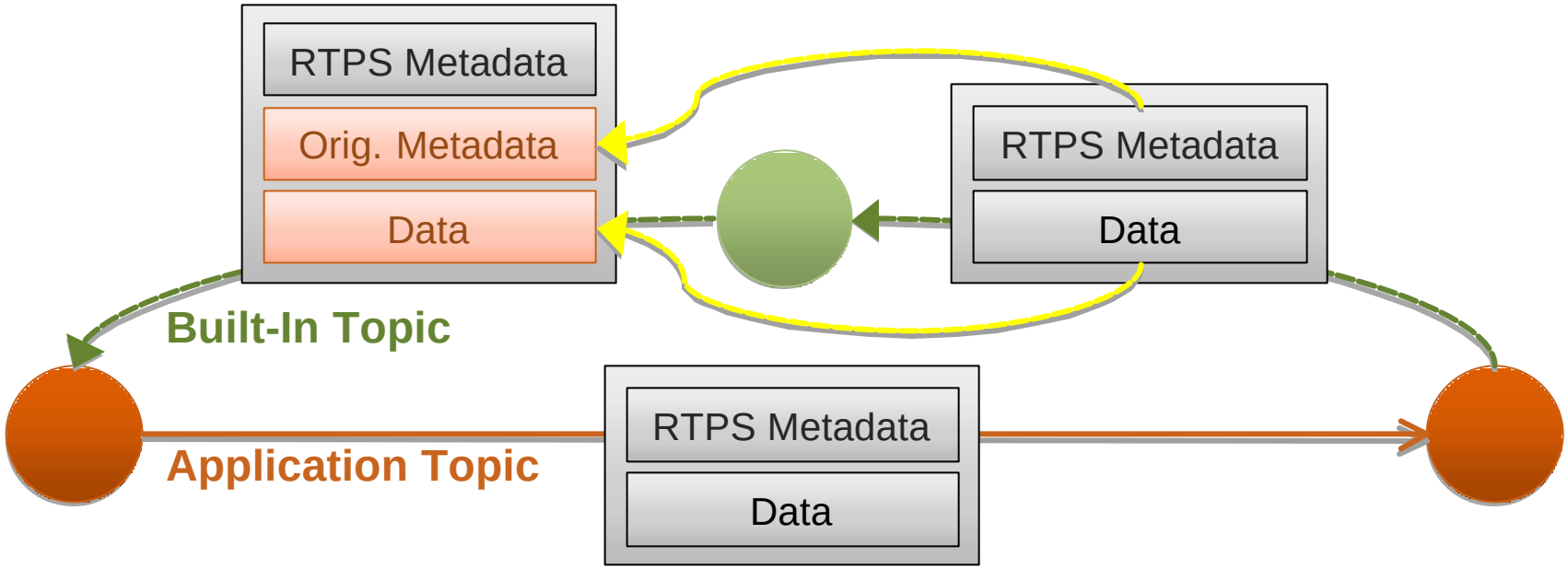
n < m

1. **Brokered discovery**



DDS Scalability: Brokered Discovery

- Can be built with RTPS-standard building blocks



RTPS `InfoSource` submessage provides data forwarding capability:
"Treat the following as if it came from that guy instead of from me."

Future Direction: Enhanced Built-in Security

- **System of systems may have non-uniform trust**
  - Multiple communities of interest w/ different entitlements/authorization
  - Infrastructure components may (not) be trusted (e.g. brokers, daemons, persistence, recording/playback, etc.)
  - *There is no "system high"*
- **Desirable new specifications**:
  - Topic-level access control
  - Sample-level tagging and labeling
  - Sample-level signing and/or encryption
- *More on security concerns in other talks this week*

Future Direction: Standardized WAN Support

- **Site-to-site comms need DDS across WAN(s)**
  - Including untrusted networks
  - Including firewall, NAT traversal
- **RTPS (DDSI) protocol designed for layering** atop diverse lower-level protocols
  - *Proven*: UDP, TCP, serial, switched fabric, Infiniband, VME, shared memory…
  - *Blessed for interop by OMG*: UDP
  - *Challenge*: UDP may not be routable, especially if multicast
- **Additional protocol mappings (PSMs)** could provide improved site-to-site interop
  - TCP for WAN routing
  - (D)TLS for transport-level security (a clarification, not a new PSM)
  - *Could be standardized quickly*

Recap

- **Large systems frequently composed** of smaller subsystems
  - Developed independently
  - …With different network environments
  - …And different data interest/entitlements

- **Best expressed in hierarchical system design**
  - Subsystems "gated" by DDS router/gateway
    - Broker internal ←→ external protocols (may be different)
    - Transform/cleanse internal ←→ external data types
    - Impose governance: enforce policy, attach monitoring
  - Inter-router integration space is itself a "subsystem" for purposes of hierarchical composition

Recap

- **DDS already provides** much of what's needed
  - Strong SLAs for governance
  - Multiple topologies for fault-tolerant subsystem isolation
  - Flexible, scalable interoperability protocol

- **Promising future directions:**
  - RTPS/TCP protocol layering for WAN traversal
  - Enhanced security model
    - Topic-level access control
    - Sample-level tagging, signing, and encrypting