Publish/Subscribe in an MQ network

(including all the new V8 bits)

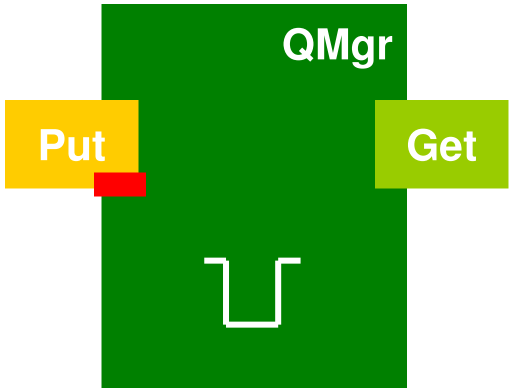
David Ware

Agenda

- The basics
- The possibilities
- Compare and contrast
- Scenarios

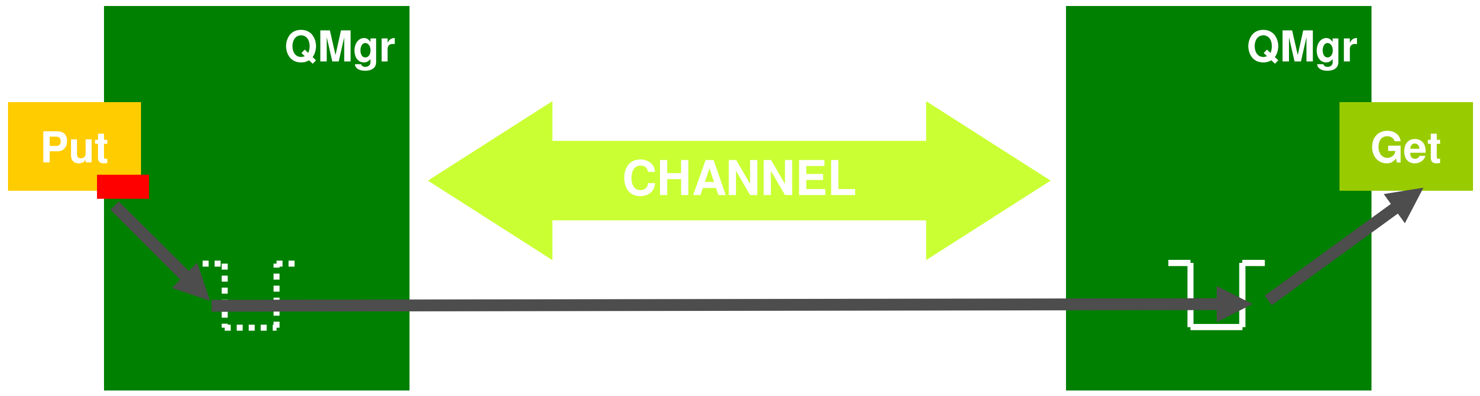
Distributed queuing

- We all know how point-to-point works…



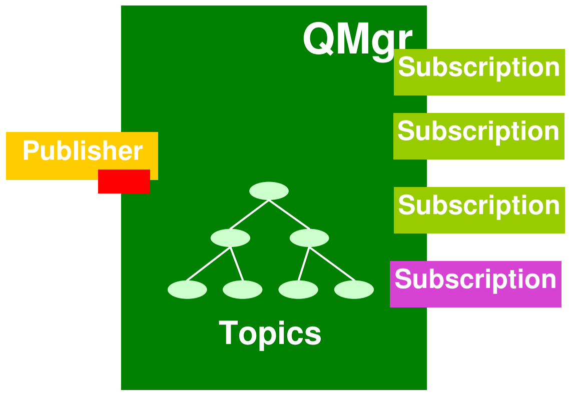
Distributed queuing

- We all know how point-to-point works
- Even across multiple queue managers



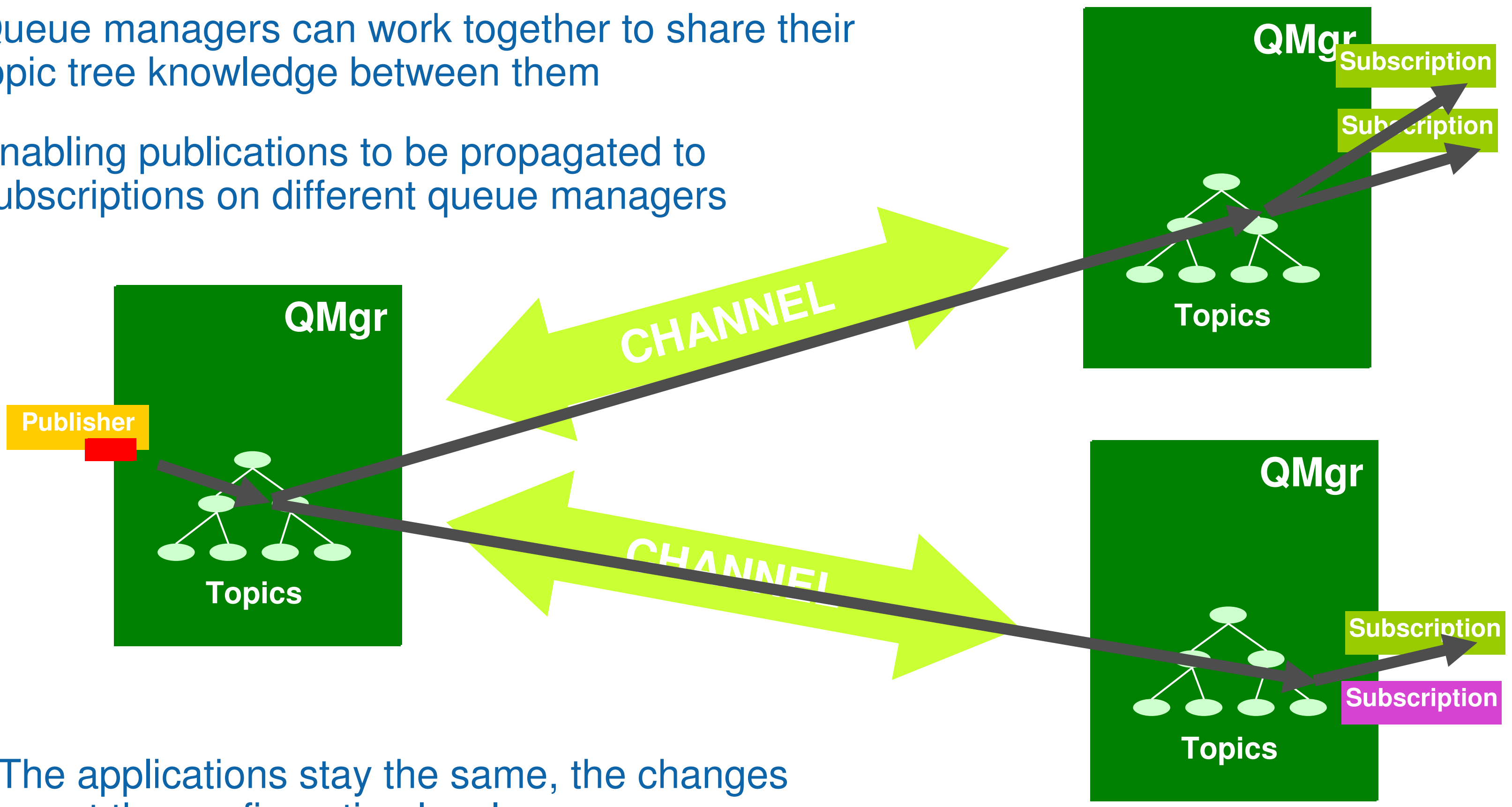
Distributed publish/subscribe

- And we know how everything revolves around the topic tree, dynamically built up in a queue manager



Distributed publish/subscribe

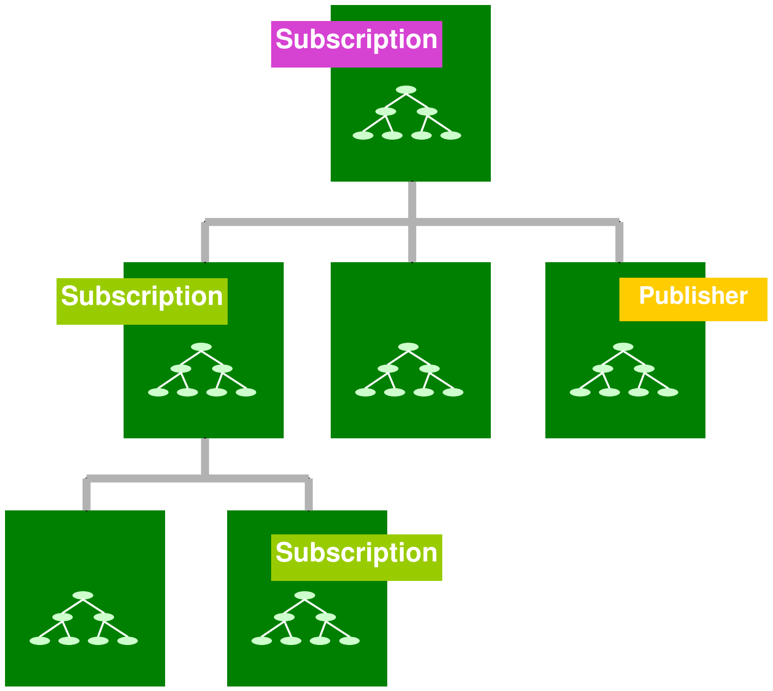
- And we know how everything revolves around the topic tree, dynamically built up in a queue manager
- Queue managers can work together to share their topic tree knowledge between them
- Enabling publications to be propagated to subscriptions on different queue managers

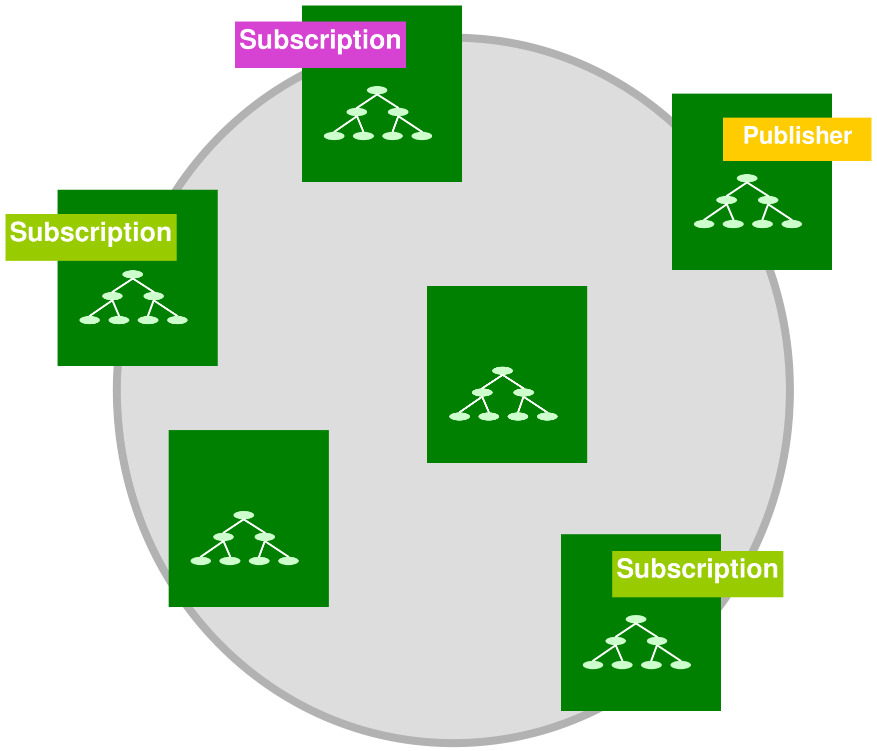


- The applications stay the same, the changes are at the configuration level.

Distributed publish/subscribe topologies

- Publish/subscribe topologies can either be created as a defined ***hierarchy***
- or more dynamically as a ***cluster***
- In both cases, publish/subscribe configuration is required…

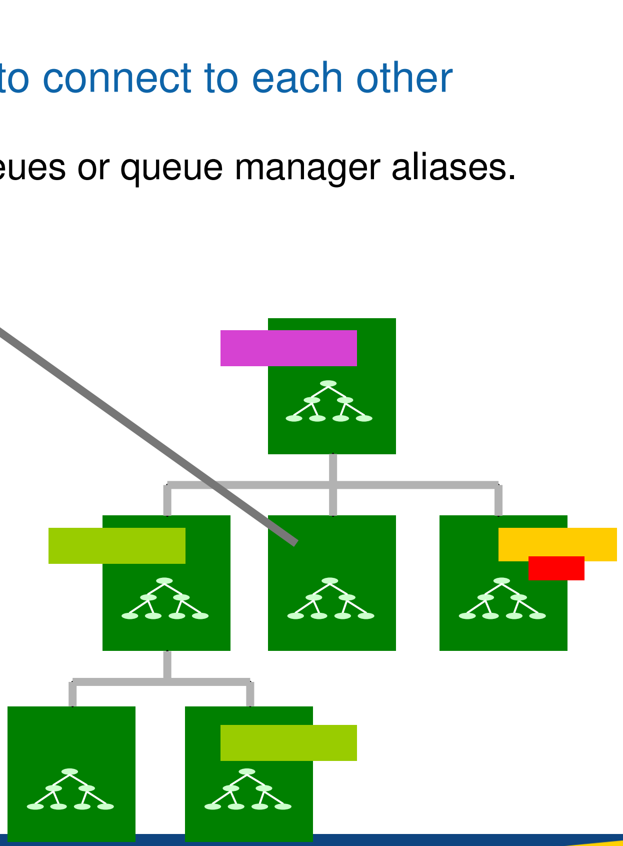




Configuring

Hierarchies

- Hierarchies have been supported since the original WebSphere MQ publish/subscribe broker support.
- Every queue manager defines its *parent* in the hierarchy.
  - (except for the root queue manager)
  - ***ALTER QMGR PARENT(QMGR1)***
- Each pair of directly related queue managers must be able to connect to each other using WebSphere MQ channels.
  - Manual channels, with queue manager named transmission queues or queue manager aliases.
  - They are both in the same cluster.
- Use **DISPLAY PUBSUB** to check the relationships
- By default all subscription knowledge is shared across the hierarchy.
  - No requirement to define administered topic objects or modify applications.
  - Controlled by scoping down topics.
    - ***Subscription Scope*** and ***Publication Scope***.



Hierarchies

- Once the *parent* relationship has been configured on the child queue manager, both queue managers should show each other when displaying PUBSUB status:

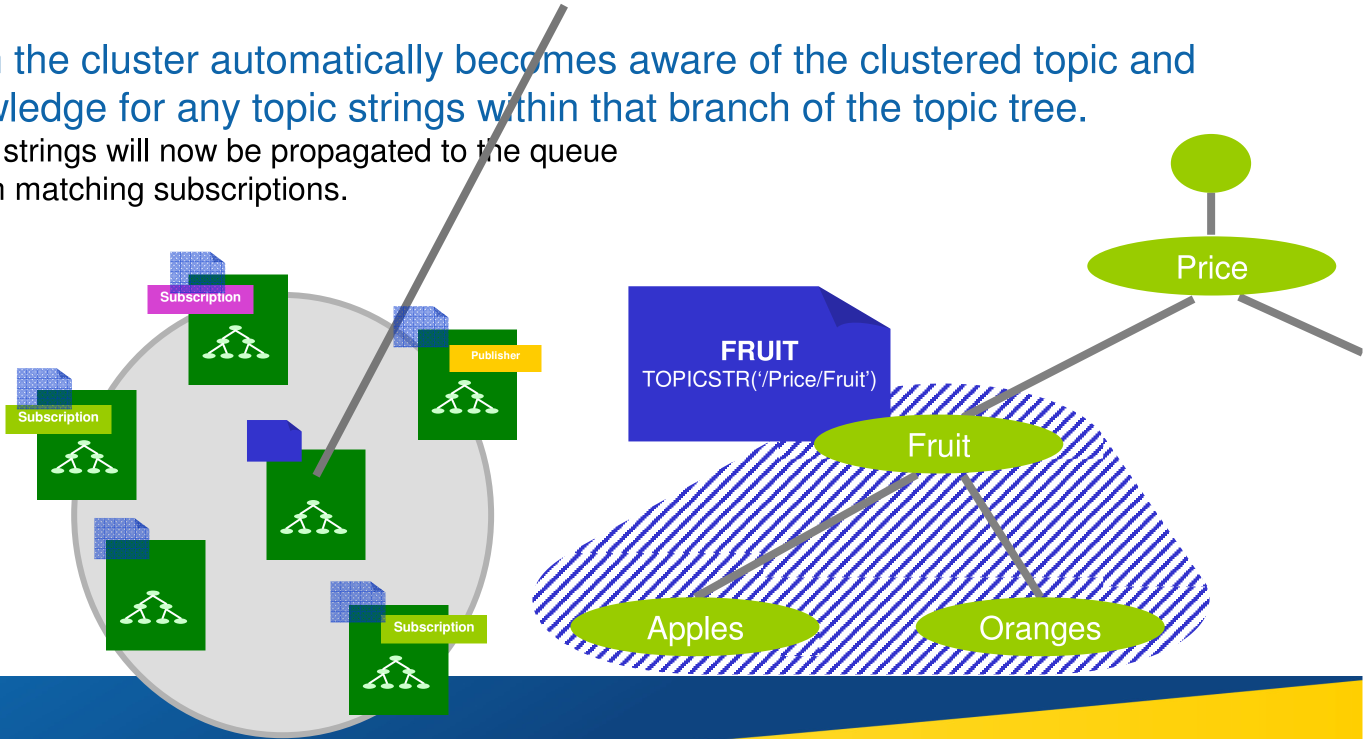
```
DISPLAY PUBSUB
     1 : DISPLAY PUBSUB
AMQ8723: Display pub/sub status details.
   QMNAME(QMB)                             TYPE(LOCAL)
AMQ8723: Display pub/sub status details.
   QMNAME(QMD)                             TYPE(CHILD)
AMQ8723: Display pub/sub status details.
   QMNAME(QMA)                             TYPE(PARENT)
```

- Additionally, informational messages will be written to each queue manager’s logs when a relationship is established:

```
AMQ5964: Pub/sub hierarchy connected.
EXPLANATION: A pub/sub hierarchy connection has been established with child or parent queue manager 'QMA'.
```

Publish/Subscribe Clusters

- No additional connectivity configuration is required (that's handled by the cluster)
- By default, no subscription knowledge is shared between members of a cluster.
  - Meaning no publications automatically flow between queue managers.
- Enabling distributed pub/sub in a cluster is initially achieved by ***clustering*** an administered topic definition on ***one*** of the queue managers in the cluster.
  - Set the cluster name to the name of the cluster.
  - ***DEFINE TOPIC(FRUIT) TOPICSTR('/Price/Fruit') CLUSTER(CLUSTER1)***
- ***Every*** queue manager in the cluster automatically becomes aware of the clustered topic and shares subscription knowledge for any topic strings within that branch of the topic tree.
  - Publications for those topic strings will now be propagated to the queue managers in the cluster with matching subscriptions.
  - No application changes are required.



NOTES

Clusters

- Once a topic object has been added to a cluster every queue manager should have visibility of it. This can be seen by displaying cluster topics on a queue manager:

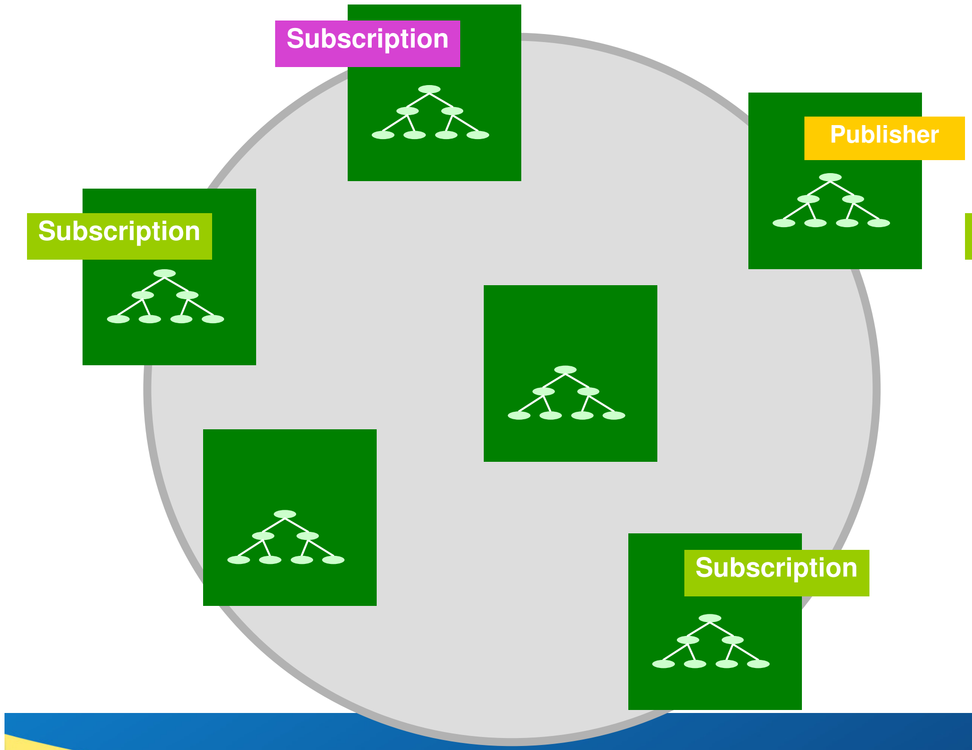
```
DISPLAY TOPIC(*) TYPE(CLUSTER) TOPICSTR CLUSQMGR CLROUTE CLSTATE
     3 : DISPLAY TOPIC(*) TYPE(CLUSTER) TOPICSTR CLUSQMGR CLROUTE CLSTATE
AMQ8633: Display topic details.
   TOPIC(FRUIT)                            TYPE(CLUSTER)
   TOPICSTR(/Price/Fruit)                  CLUSQMGR(QMGR2)
   CLROUTE(DIRECT)                         CLSTATE(ACTIVE)
```

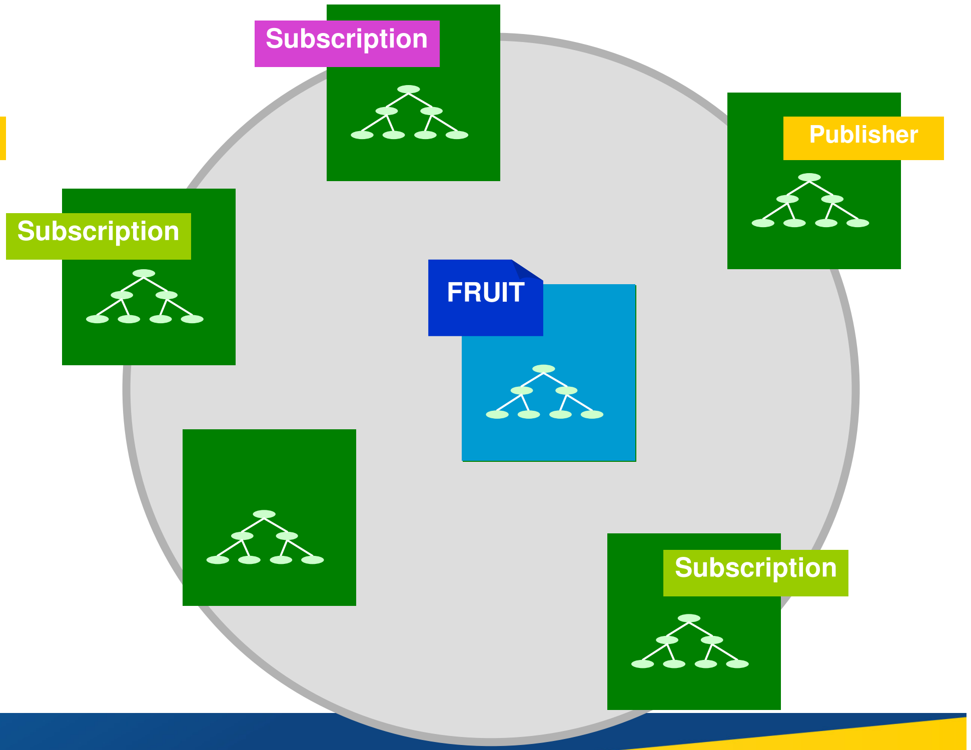
- The above display command includes CLROUTE and CLSTATE. These are new for **WebSphere MQ V8**.
  - CLROUTE is covered in the following slides.
  - CLSTATE gives feedback on the current state of this cluster object. Anything other that ACTIVE implies a problem that should be investigated.
- If the cluster topic is not visible on a queue manager, check that the cluster channels too and from the full repositories are functioning correctly and that no errors are reported on those full repositories, the queue manager where the topic is defined or the queue manager where the topic was not visible.

Clusters

V8

- Publish/subscribe in clusters now has two modes of routing publications.
  - **DIRECT**: Publications are sent directly from the publisher's queue manager to every queue manager in the cluster with a matching subscription.
  - **TOPIC HOST**: Publications are routed via one or more queue managers in the cluster. These are the queue managers where the clustered topic definitions were defined.
- *DEFINE TOPIC … CLUSTER(…) CLROUTE(DIRECT|TOPICHOST)*





Cluster routing

- WebSphere MQ V8 introduces the new topic object attribute of CLROUTE *(ClusterPubRoute)*. This takes two values:
  - DIRECT
    - All queue managers in the cluster are aware of everyone else and will send publications directly to other queue managers in the cluster with matching subscriptions. This is the default.
  - TOPICHOST
    - Only the queue manager(s) where the topic object is defined are aware of everyone else. Publications are forwarded to these topic hosting queue managers, who then forward onto queue managers with matching subscriptions.
- To use topic host routing in a cluster, the following queue managers must be at V8 or above:
  - The topic hosting queue managers
  - The full repository queue managers
  - The queue managers where subscriptions exist
  - The queue managers where publishers connect
- Any older queue managers will not be aware of the topic host routed topic definition and continue to behave as if it was not defined in the cluster.

Which is best?

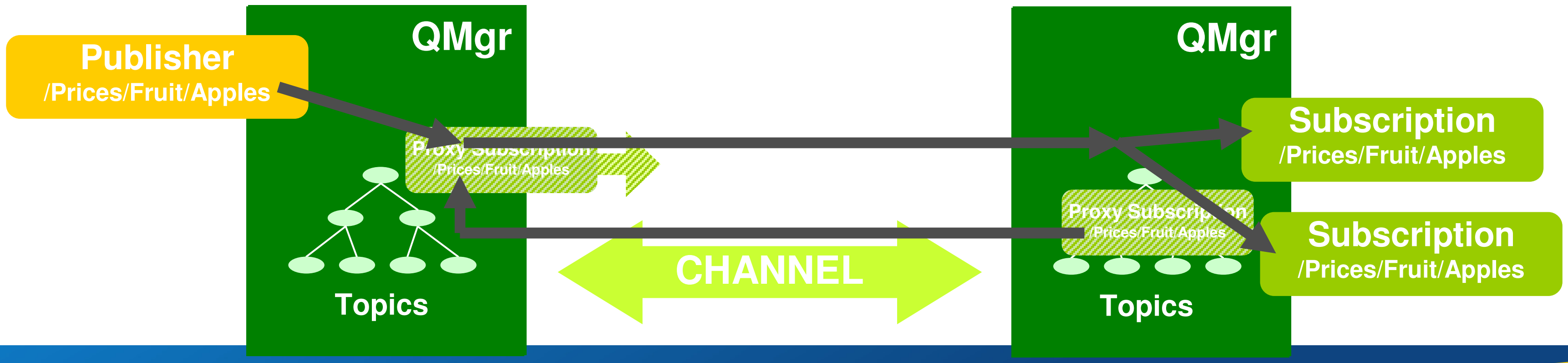
- Should I use a ***hierarchy***, a ***direct*** or a ***topic host cluster topic***?
- *It depends…*

- Many factors will influence the design of a pub/sub topology:
  - Number of queue managers in the topology
  - Size and shape of those queue managers
    - Their capacity, availability, lifetime, interconnectivity…
  - Number and lifetime of subscriptions
  - Number and lifetime of individual topic strings
  - Publication traffic
  - Distribution pattern of publications to subscribers
  - …
- One size does ***not*** fit all.
- First, you need to understand more about how it works…

The mechanics

How it works: subscription propagation

- A subscription for a topic string is created on a queue manager.
- That queue manager tells others that it is interested in that topic string.
- Those queue managers register ***proxy*** subscriptions to represent the interest.
- On a queue manager, when a message is published to the topic string, a copy of the message is sent, over WebSphere MQ channels, to each queue manager that a proxy subscription is held for.
- The receiving queue manager processes the message and gives a copy to each subscription it holds *(including any proxies…).*
- Additional subscriptions on the same topic string do not require additional proxy subscriptions to be flowed.



Proxy subscriptions

- Proxy subscriptions are a special type of subscription that tells one queue manager that another queue manager needs a copy of any matching publications to be sent to it.
- Proxy subscriptions can be viewed on a queue manager using MQSC or MQ Explorer. This shows which queue managers will receive a publication for which topic.

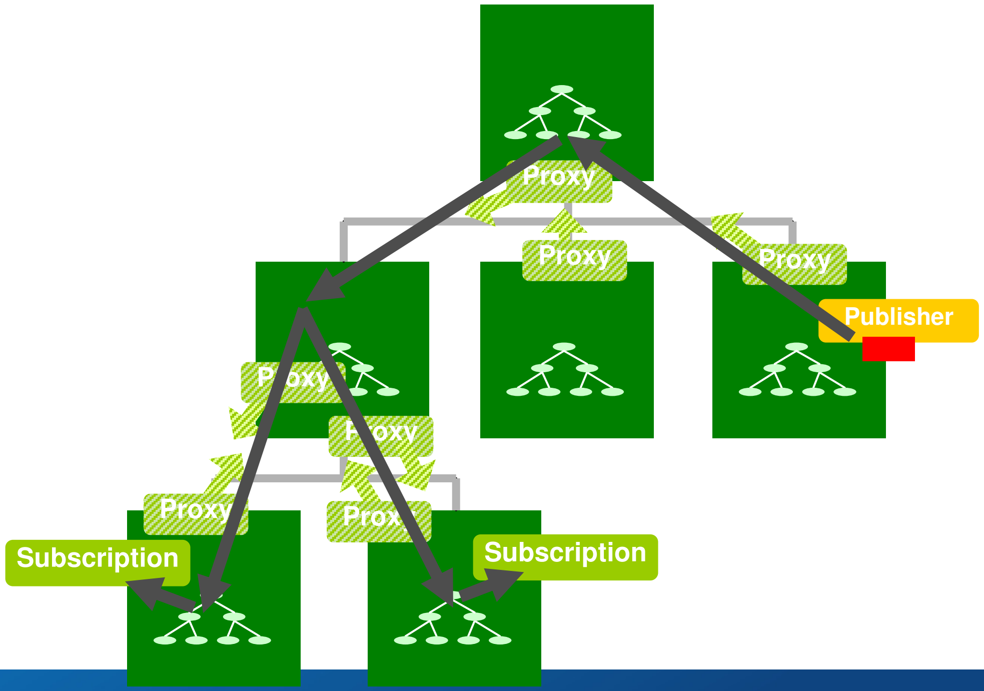
```
DISPLAY SUB(*) SUBTYPE(PROXY) TOPICSTR DESTQMGR

AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120514D47523120202020202020202021123C5320000E02)
   SUB(SYSTEM.PROXY.QMGR2 CLUSTER1 /X/Y)   TOPICSTR(/X/Y)
   DESTQMGR(QMGR2)                         SUBTYPE(PROXY)
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120514D47523120202020202020202021123C5320000E05)
   SUB(SYSTEM.PROXY.QMGR2 CLUSTER1 /X/Z)   TOPICSTR(/X/Z)
   DESTQMGR(QMGR2)                         SUBTYPE(PROXY)
```

- If a queue manager has multiple subscriptions to the same topic string only a single proxy subscription is generated.
- Proxy subscriptions do not include any selectors set on the subscription.

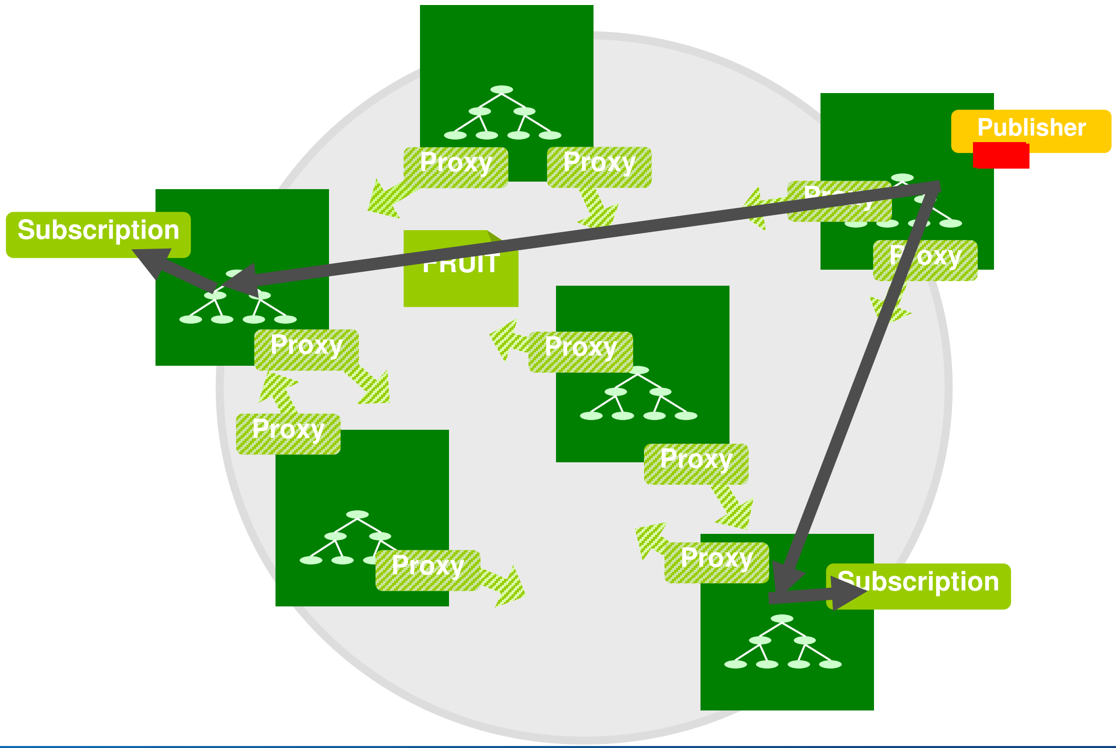
How it works: in a *hierarchy*

- Proxy subscriptions are sent from a queue manager to every ***directly*** connected queue manager in the hierarchy.
- They in turn send proxy subscriptions onto any further relations.
- Until everyone in the hierarchy is aware of the topic string being subscribed to.
- Publications are then sent back down the path of proxy subscriptions.



How it works: *Direct* routed clustered topics

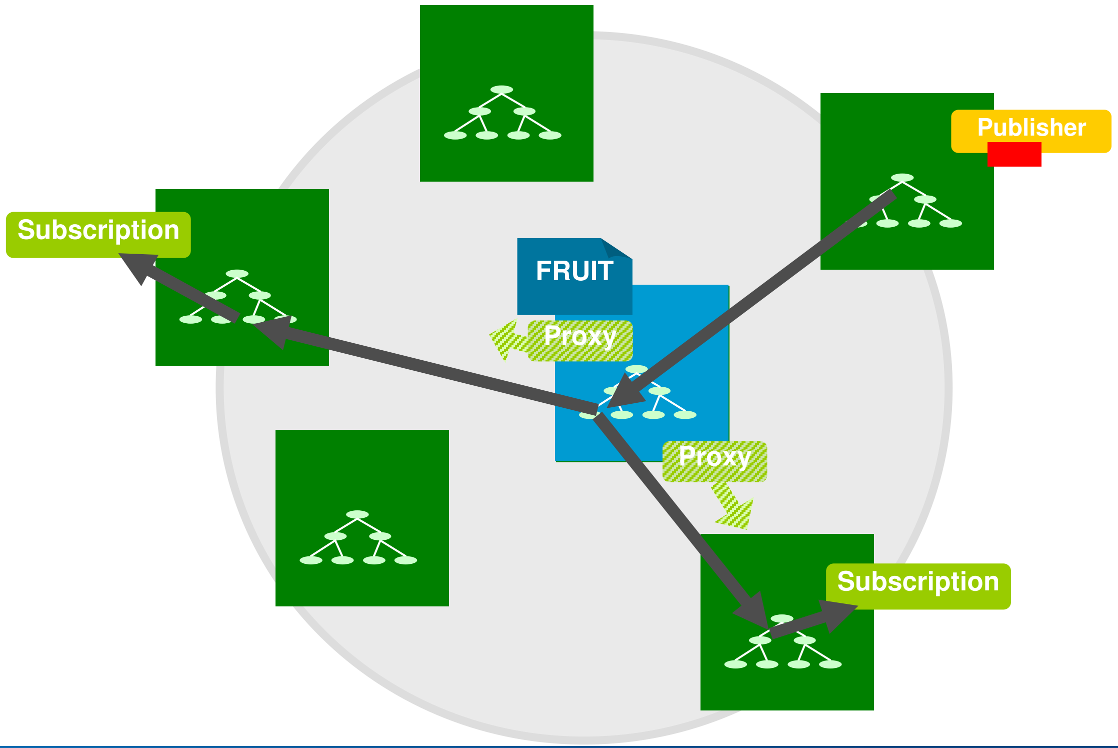
- Proxy subscriptions are sent from a subscribing queue manager directly to ***every*** other queue manager in the cluster.
  - *Every queue manager becomes aware of everyone else in the cluster.*
- Any publication from a queue manager is sent directly to those queue managers which sent proxy subscriptions.



V8

How it works: *Topic host* routed clustered topics

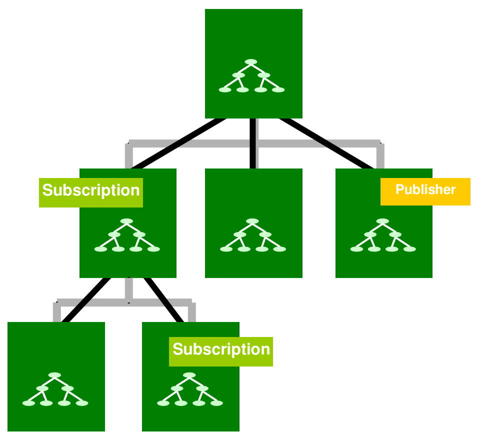
- Proxy subscriptions are sent from a subscribing queue manager ***only*** to queue managers that host a definition of the clustered topic.
  - Only the topic hosts are aware of everyone else.
- Any publication from a queue manager is ***always*** sent to one of the topic hosts, who then forwards the message to any subscribing queue managers.



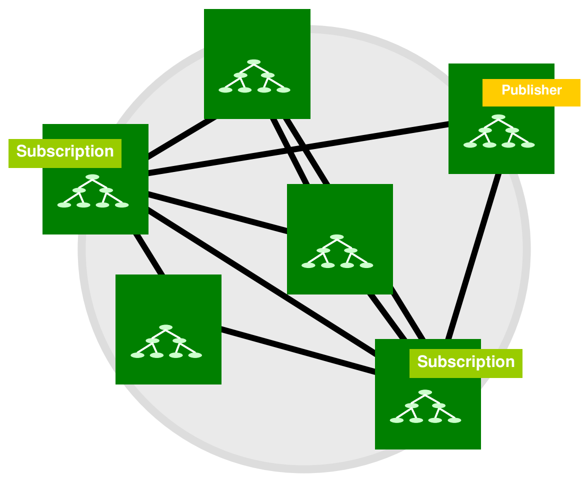
Side-by-side

Side-by-side: scaling

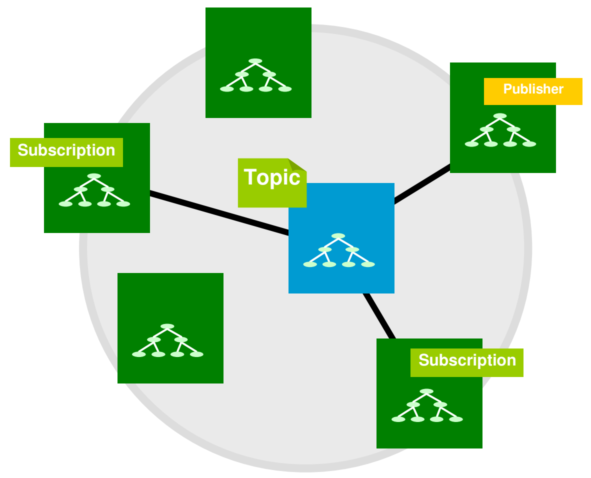
- Channel connectivity is restricted in both topic host routed clusters and hierarchies.
- Direct routed clusters will result in many queue manager channels being established, even to queue managers with no publishers or subscribers.
  - ***This can be a concern for large clusters***.
- Hierarchies and direct routed clusters share subscription knowledge across **all** queue managers.



Hierarchy



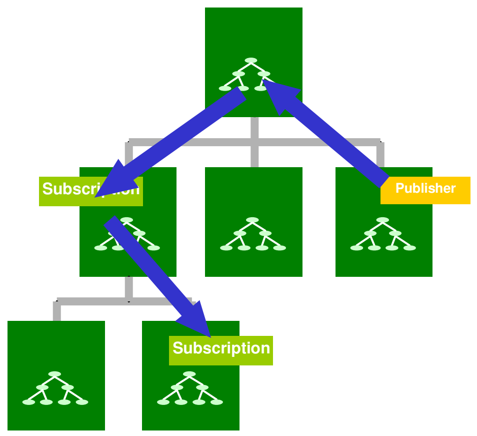
Direct Cluster



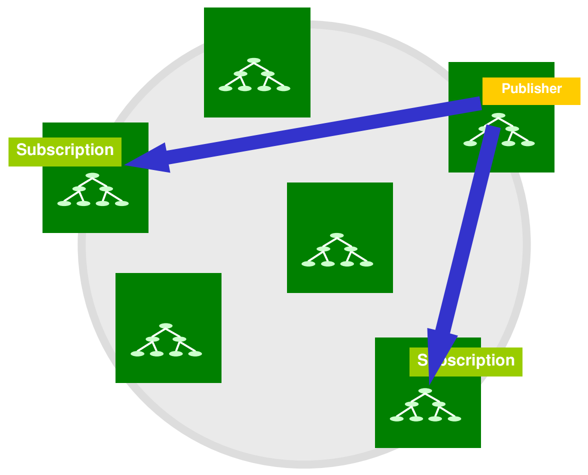
Topic host cluster

Side-by-side: publication flows

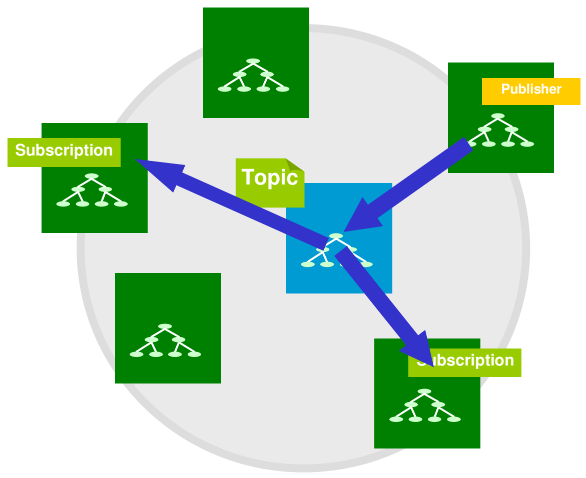
- Direct routed clusters ensure publications take the shortest path between publishers and subscribers.
- Topic host routed clusters can introduce an additional hop, with hierarchies relying on even more.
  - In both of these, careful placing of subscribers and publishers can achieve the same single-hop route that direct clusters provides (e.g. host the topic on the publisher’s queue manager in the cluster).



Hierarchy:1,2,3… hops



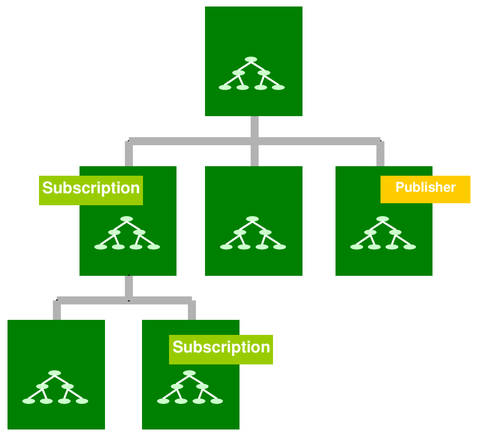
Direct Cluster: 1 hop



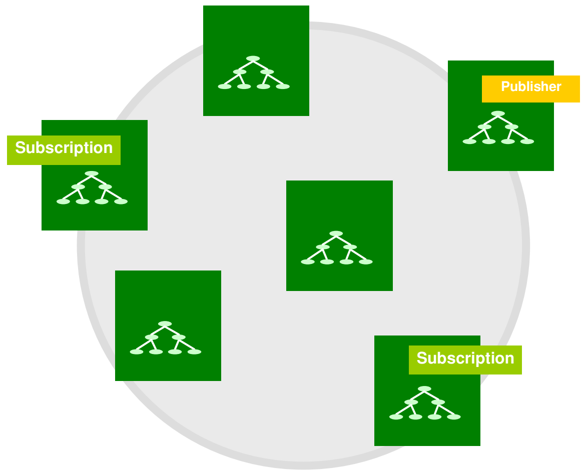
Topic host cluster: 1 or 2 hops

Side-by-side: configuration

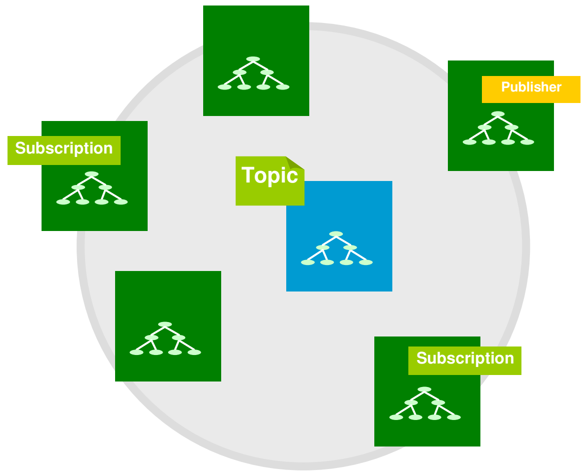
- Hierarchies require fine grain planning and configuration of queue managers.
  - Harder to alter the hierarchical structure.
- Clusters allow queue managers to join and leave without too much consideration once the overall plan is defined.
  - Careful planning of topic host routers is required.



Hierarchy



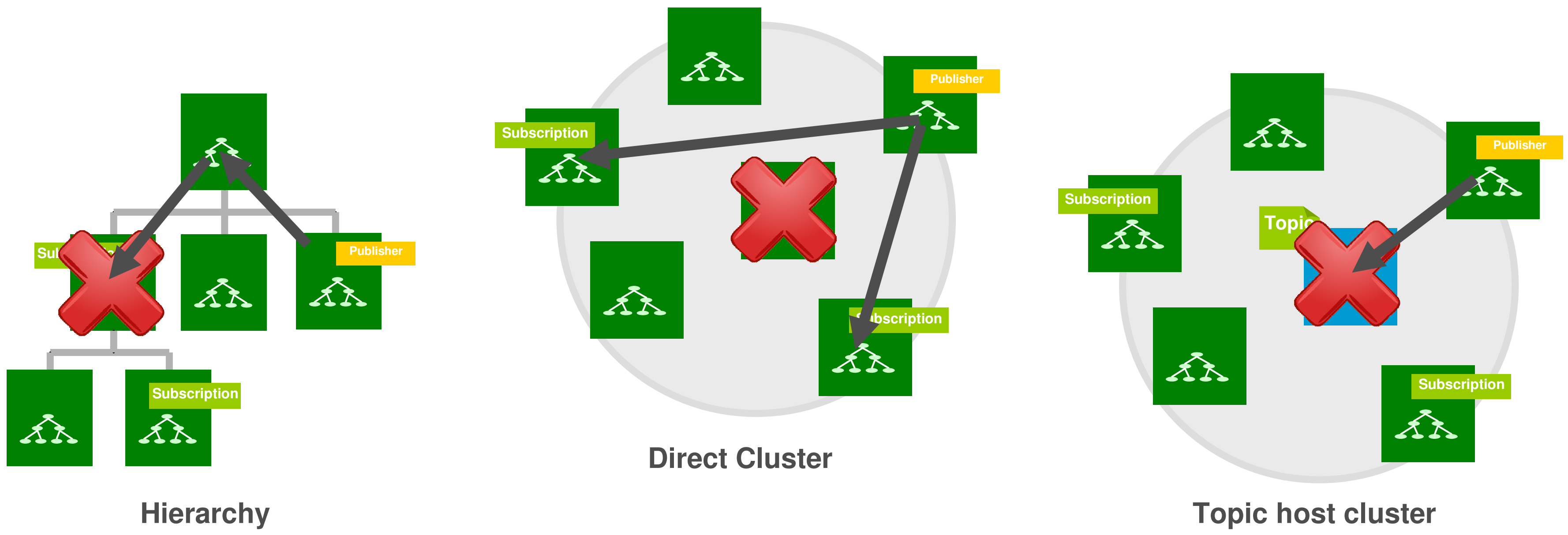
Direct Cluster



Topic host cluster

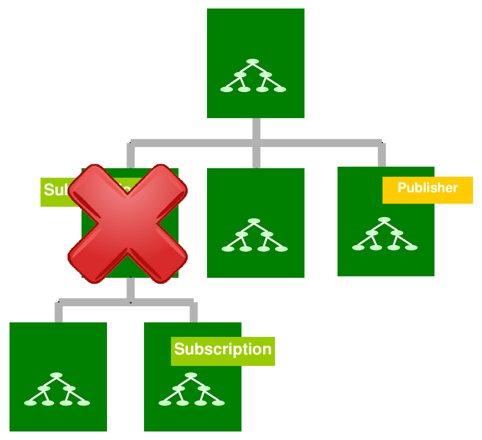
Side-by-side: availability

- An unavailable queue manager in a hierarchy can cut off many others.
- In a direct routed cluster it'll only affect the subscribers and publishers on that queue manager.
- In a topic host routed cluster it can affect more if the queue manager hosts a topic definition.
  - ***But this can be avoided…***

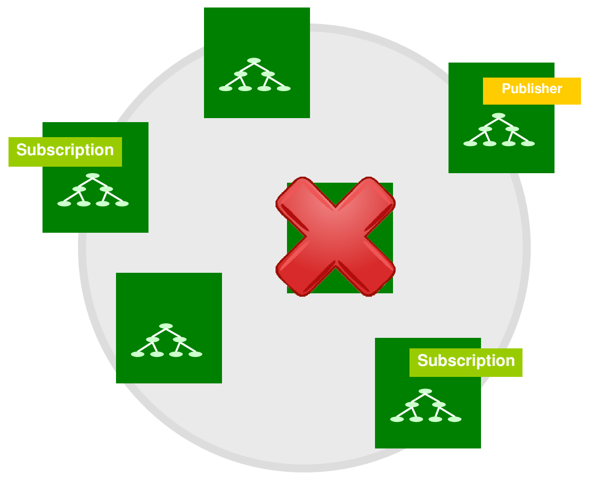


Side-by-side: availability

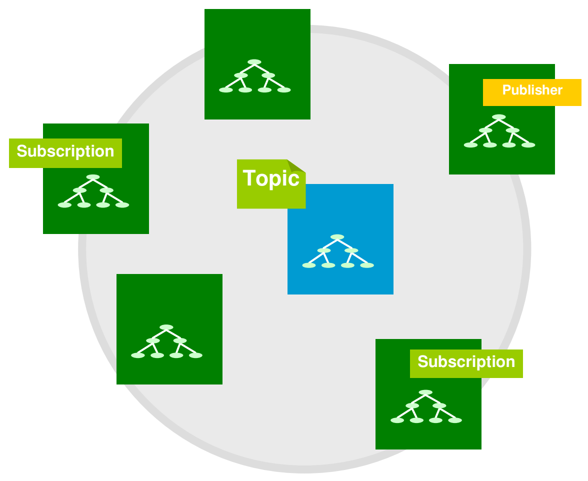
- An unavailable queue manager in a hierarchy can cut off many others.
- In a direct routed cluster it'll only affect the subscribers and publishers on that queue manager.
- In a topic host routed cluster it can affect more if the queue manager hosts topics, but this can be avoided…



Hierarchy



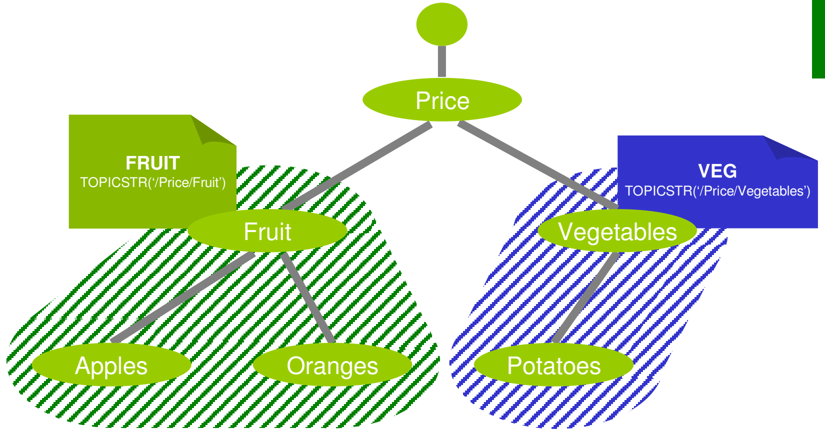
Direct Cluster

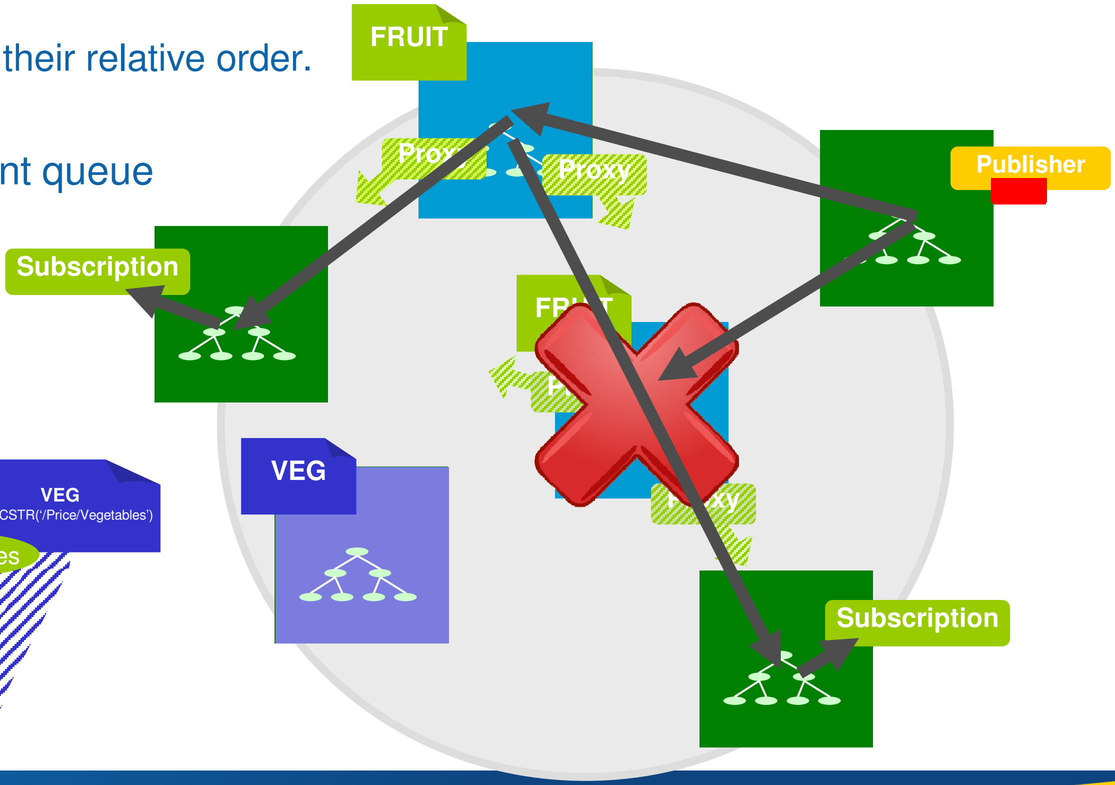


Topic host cluster

Topic host routing

- Multiple queue managers can host the *same* clustered topic.
  - With the same name, topic string, etc.
- Publications are routed based on availability and workload balancing.
- This increases availability and scalability of routes.
  - Multiple topic hosts for a topic enable workload balancing of publication traffic.
- Publications will no longer maintain their relative order.
- Hosting different topics on different queue managers can also be used to increase availability and scaling.





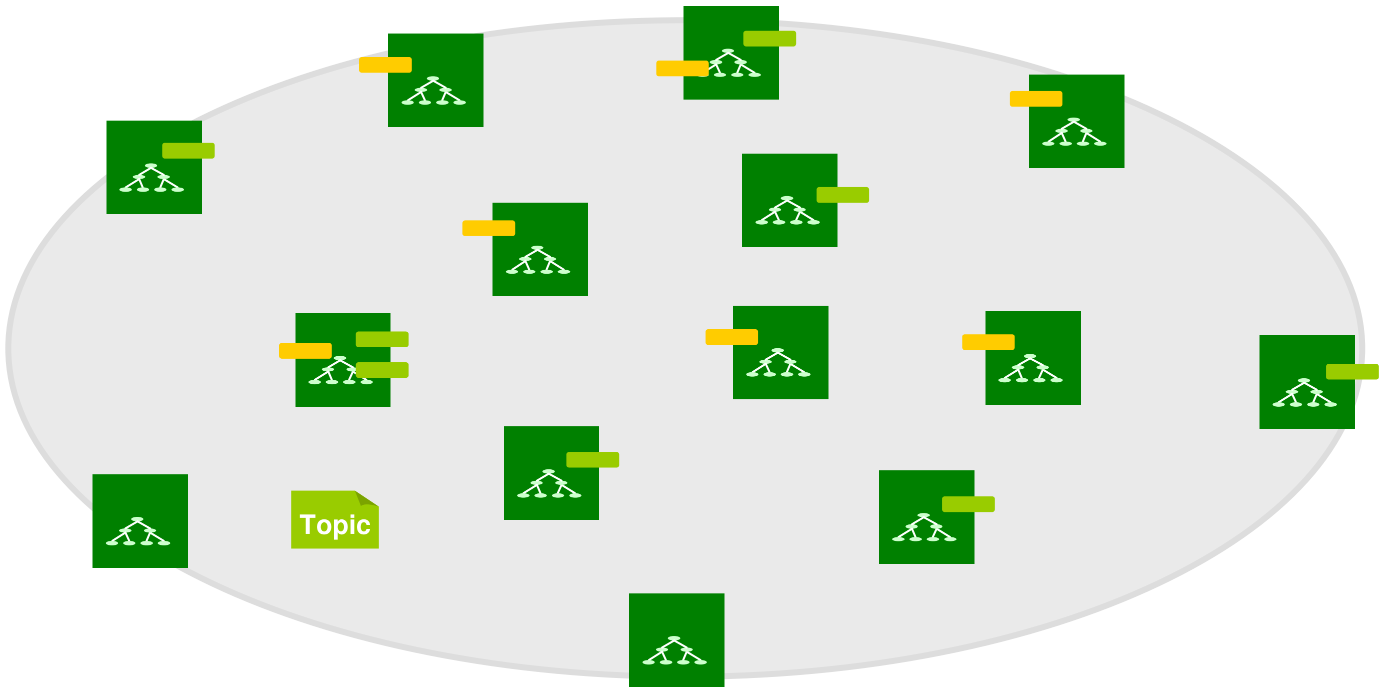
Comparison summary

- **Scaling**
  - Topic host routed clusters provide the best options for scaling.
- **Availability**
  - Clusters can be configured to avoid single points of failures, unlike hierarchies.
- **Publication performance**
  - All topologies use very similar inter queue manager techniques for transferring messages, and each can be designed to minimise routes between queue managers.
- **Configuration**
  - Clustering is the most dynamic and simplest solution – especially where clusters are already in use.

Putting it into practice

'A bit of pub/sub'

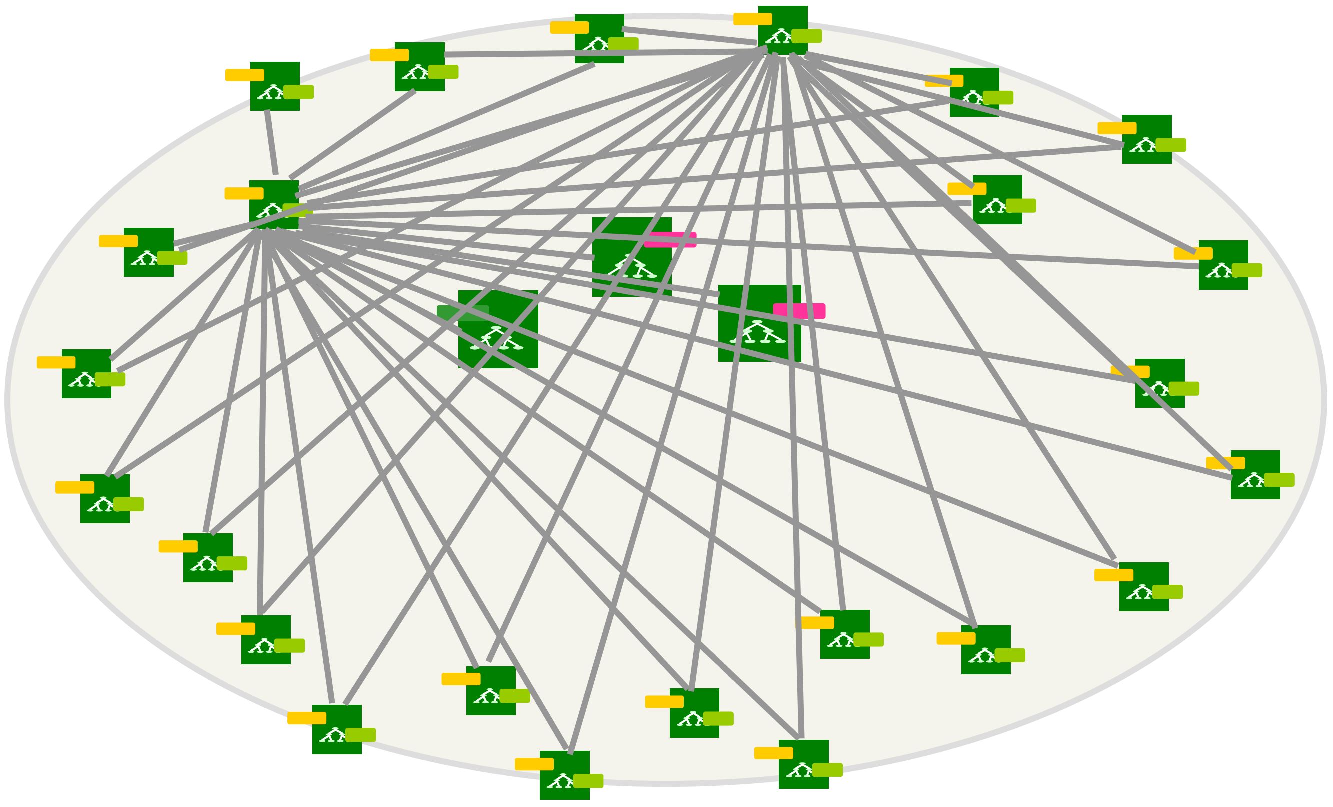
- A couple of dozen queue managers.
- Administratively created subscriptions for each major application, perhaps a hundred, with different topics, rarely changing.
- Publishers and subscriptions spread across most of the queue managers.



- The simplest to create is a cluster with **direct routed** topics.

A big ‘hub and spoke’ cluster

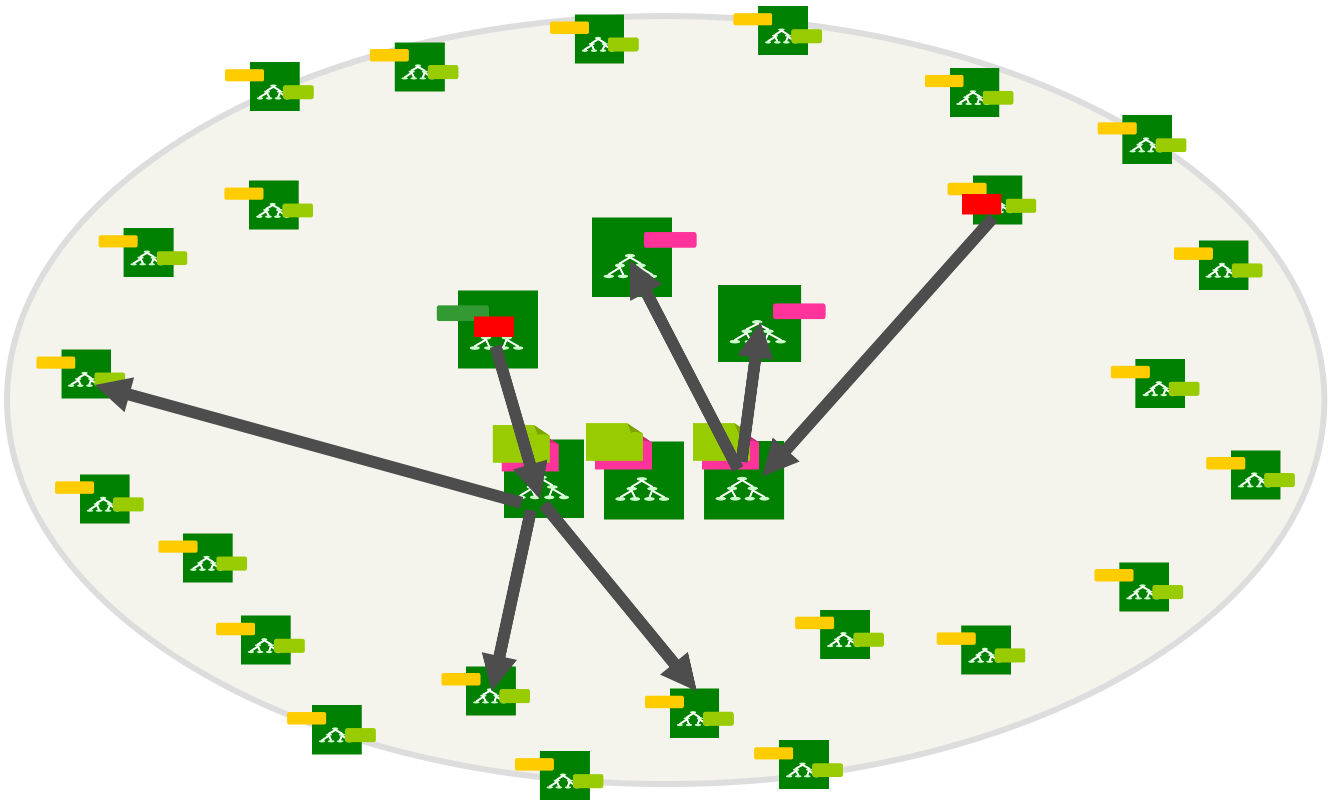
- A thousand queue managers, all talking to a few in HQ, but rarely to each other.
- Each with its own subscriptions and publishers.



- ***Do not use a direct cluster topic!***

A big 'hub and spoke' cluster

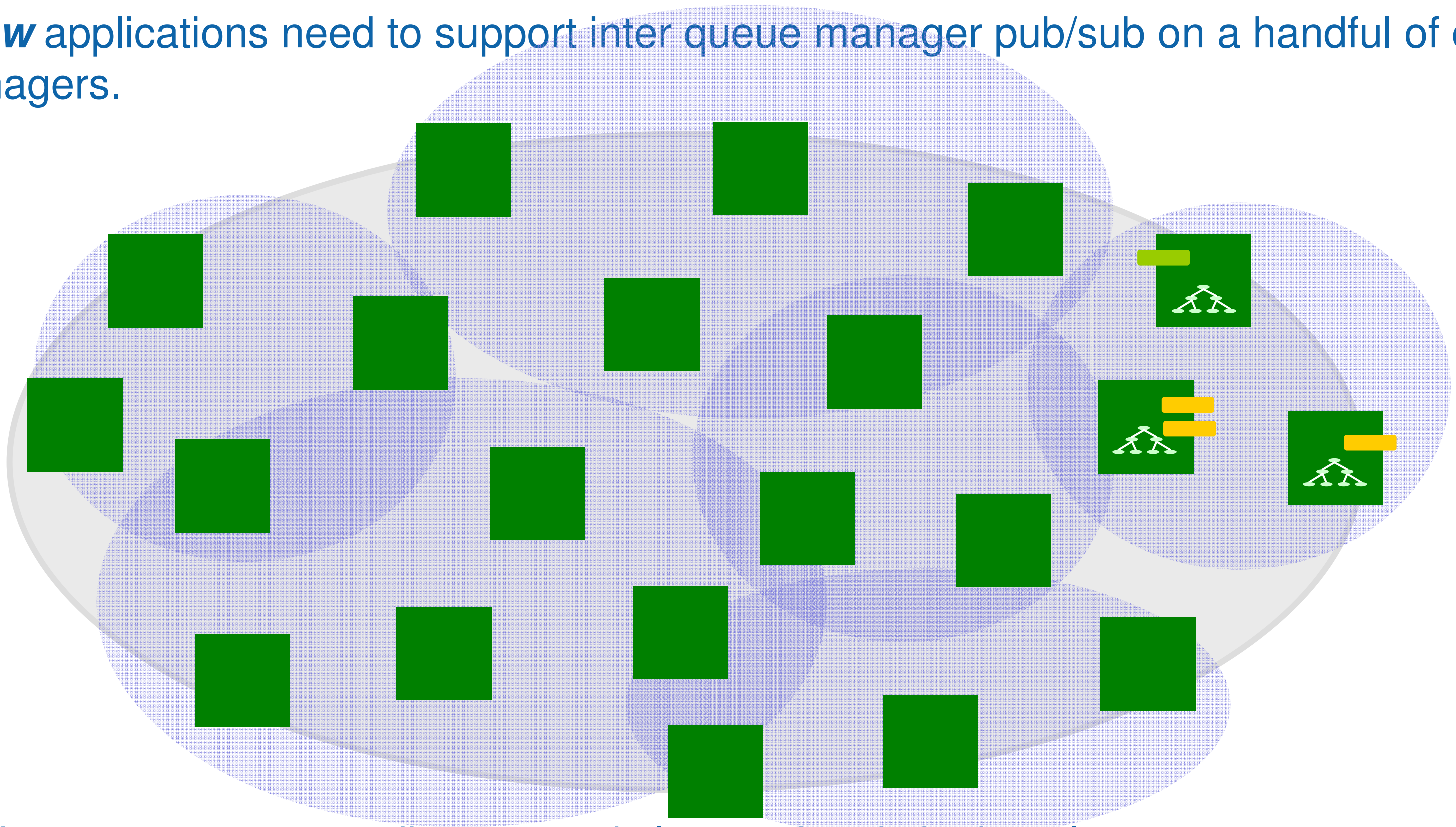
- A thousand queue managers, all talking to a few in HQ, but rarely to each other.
- Each with its own subscriptions and publishers.



- ***Do not use a direct cluster topic!***
- Use ***topic host*** routed clustered topics, possibly on a dedicated set of routing queue managers, sized to take the load.

A small amount of pub/sub within a much larger cluster

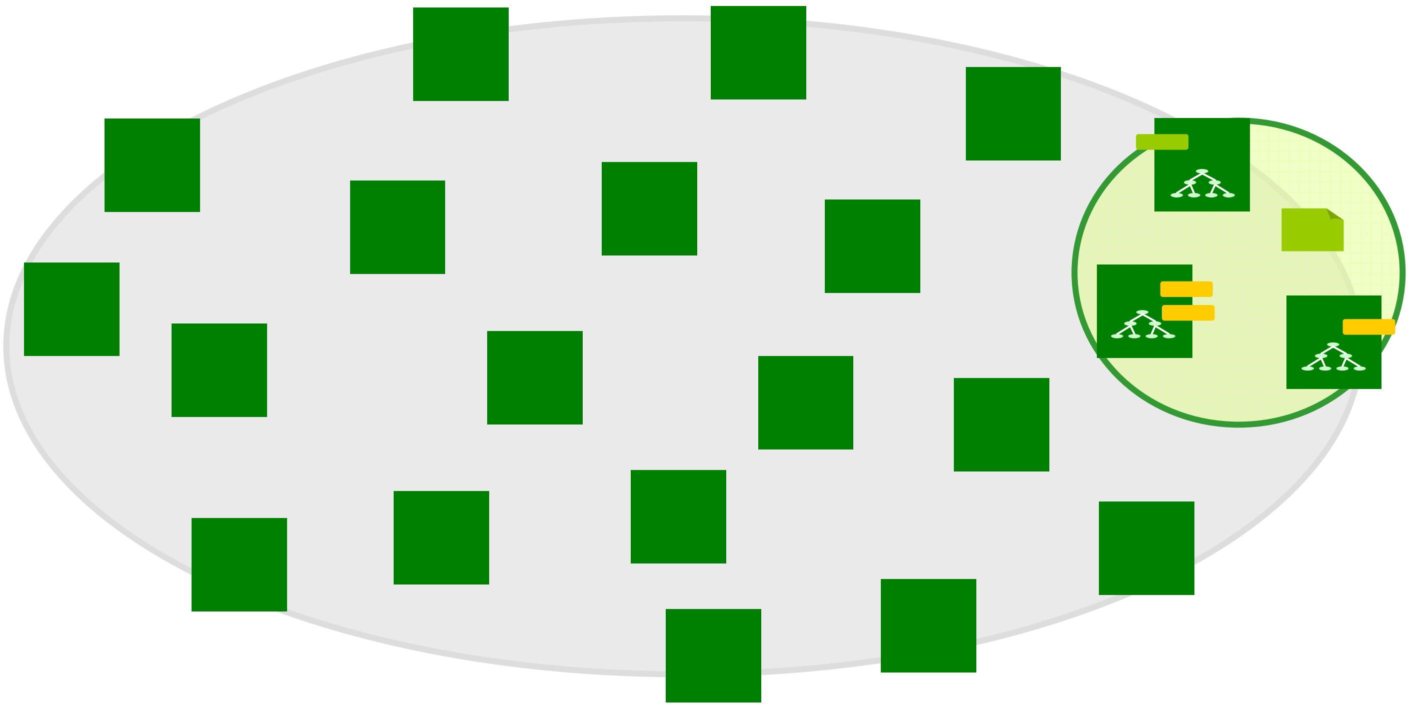
- An existing single cluster containing hundreds or thousands of queue managers. Each connected to a few others in the cluster, as required.
- A ***few*** applications need to support inter queue manager pub/sub on a handful of queue managers.



- It is best **not** to create a direct routed clustered topic in that cluster.
  - This would result in *all* queue managers learning and potentially connecting of each other.

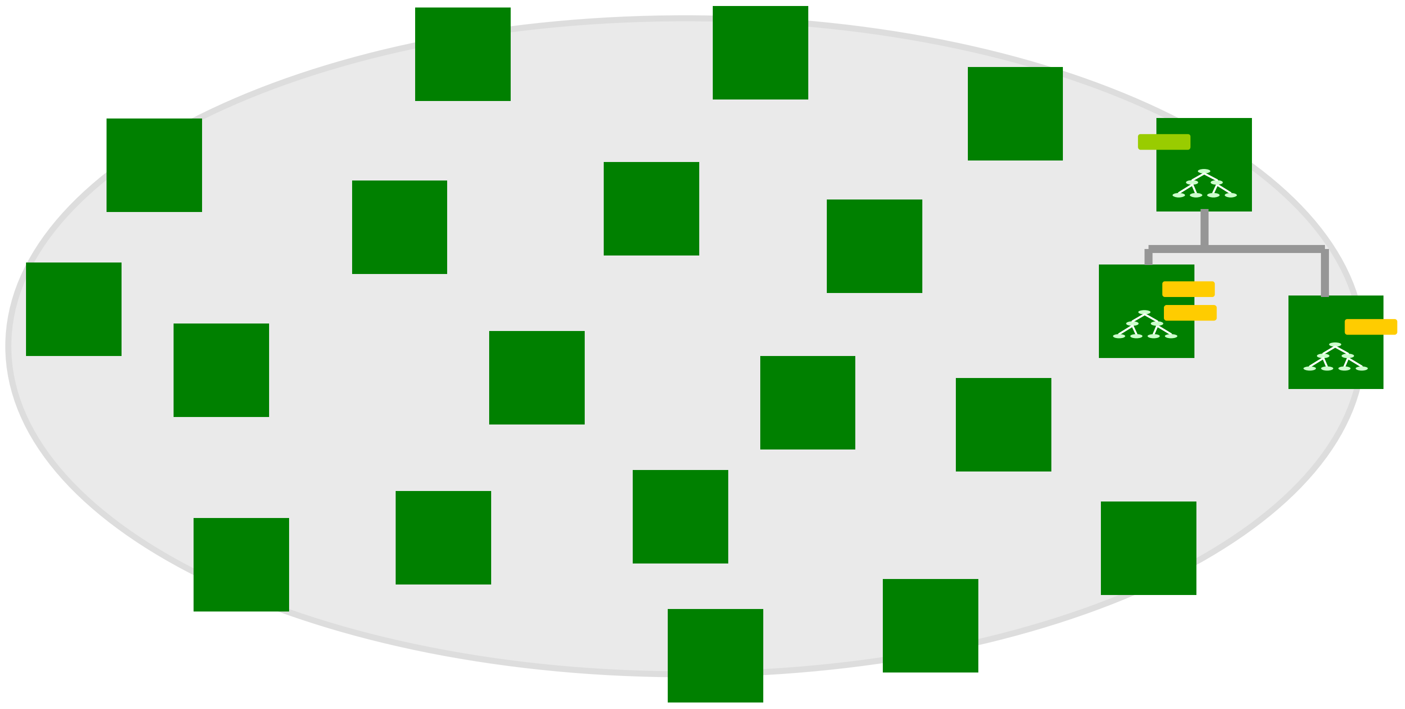
A small amount of pub/sub within a much larger cluster

- One solution is to create a dedicated overlaid cluster for the queue managers requiring publish/subscribe and clustering a **direct** topic.
  - Optionally use ***PSCLUS*** to prevent clustered topic definitions being created in the wider cluster. V7.1
    - The key is to set it on the full repositories



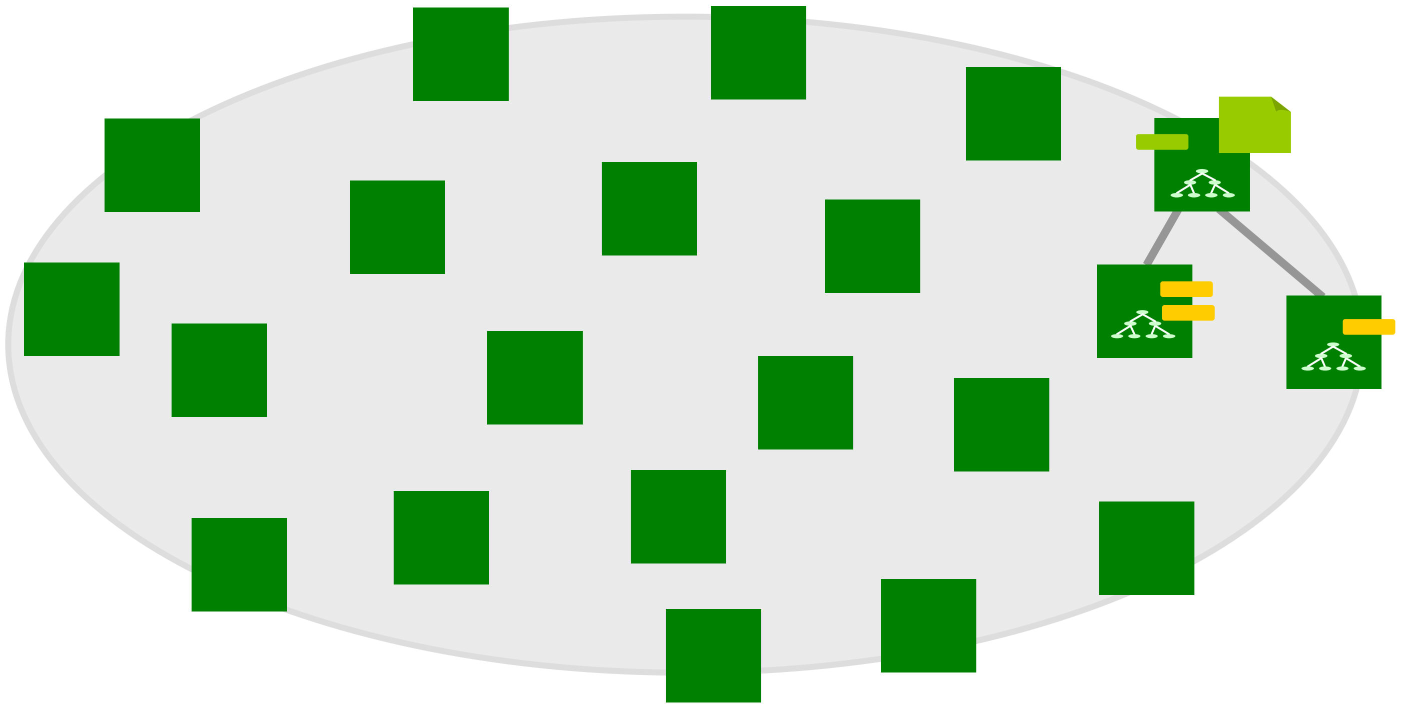
A small amount of pub/sub within a much larger cluster

- Alternatively, create a hierarchy and **don't** cluster the topic
  - Use ***PSCLUS*** to prevent clustered topic definitions being created V7.1



A small amount of pub/sub within a much larger cluster

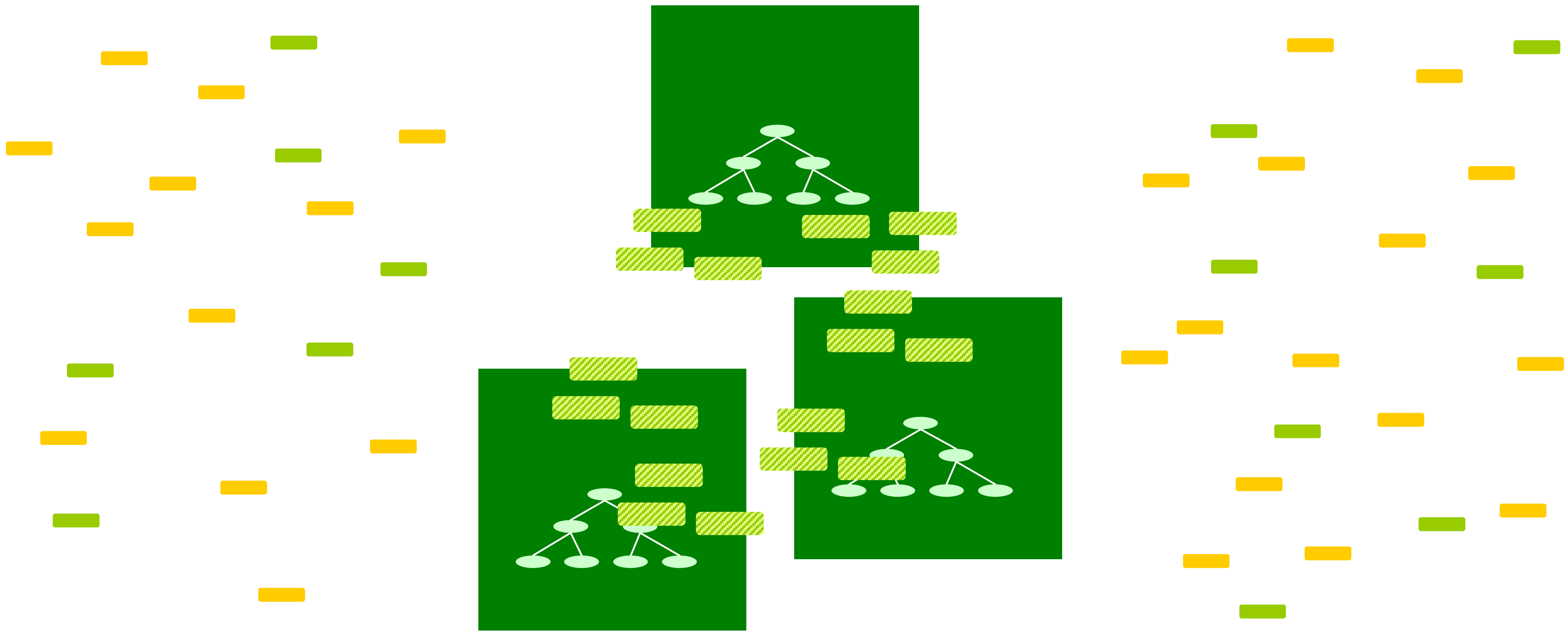
- Or use a topic host routed clustered topic.
  - All queue managers become aware of the topic but only those using it will communicate with the topic host queue manager(s).
    - *The topic hosting queue managers may need to communicate with all other queue managers under certain error circumstances.*



A very dynamic setup

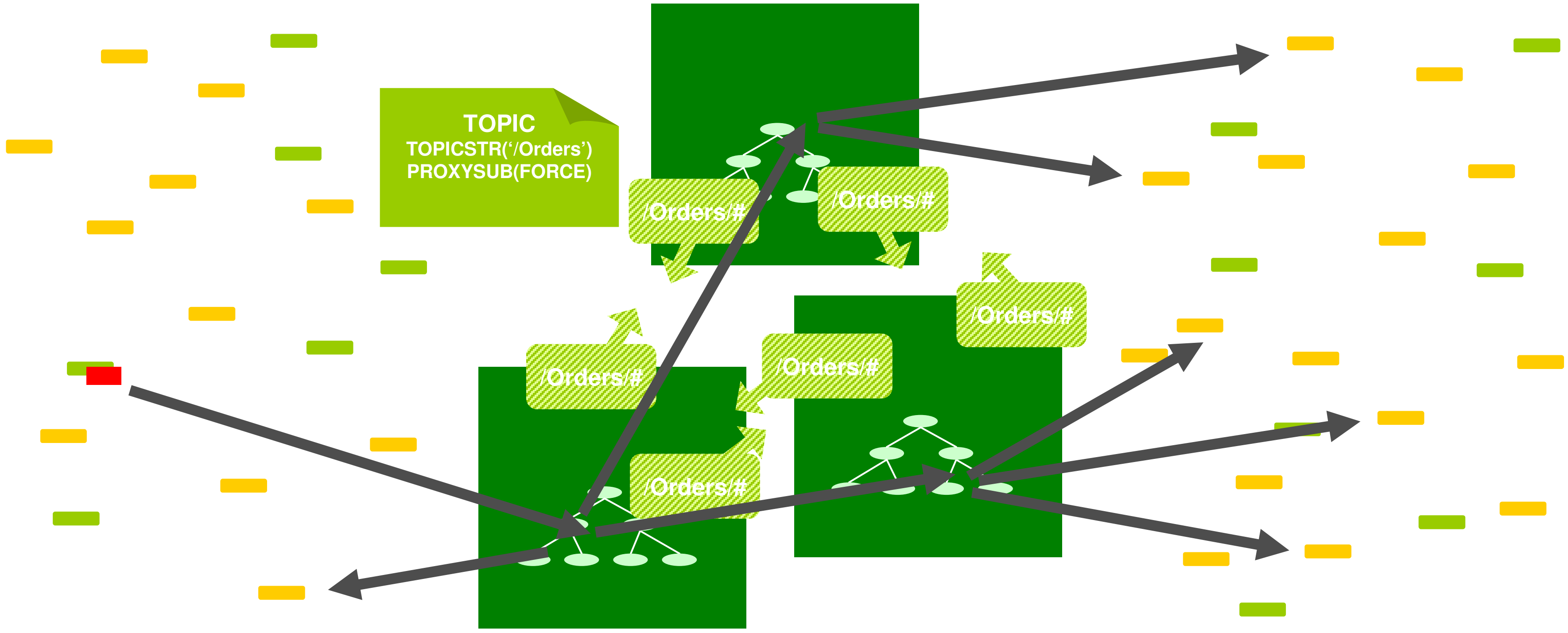
A very dynamic setup

- Thousands of loosely connected dynamic subscriptions, on thousands of different topic strings, continually coming and going, along with corresponding publishers.
- A few queue managers to provide availability and subscription load sharing.
- This introduces a special problem….
  - *Proxy subscription '**churn**'*



Publication broadcast

- Setting *proxy subscription* to ***force*** generates a ***wildcarded*** proxy subscription on all queue managers where that topic is seen.
  - All queue managers when in a cluster.
  - Only the queue manager where it is defined when in a hierarchy.
- Once that is in place there is not a need for individual proxy subscriptions for any sub-topic string.
  - *This is automatic in WebSphere MQ V8, but requires intervention prior to this* V8
- Every publication is sent to every queue manager and onto the subscriptions.
  - If no local subscriptions exist for a publication, it is simply *discarded*.



Subscription propagation or publication broadcast?

- Some scenarios mean the benefits of a **subscription propagation** model start to be eclipsed by the overheads.
  - A large set of fast changing subscriptions can result in proxy subscription churn.
- Proxy subscriptions have a per topic string granularity.
  - Proxy subscription propagation is asynchronous.
    - The delay can become a problem.
- An alternative is to **broadcast publications** across multiple queue managers on the expectation that subscriptions will exist.
  - No need to propagate the per topic string proxy subscriptions.
  - Potentially extra publication load (every queue manager sees every publication).
- Broadcast mode is enabled through the use of setting the ***Proxy Subscription*** attribute to ***Force*** on an administered topic object.

V8 *Improved in WebSphere MQ V8*

Subscription propagation or publication broadcast?

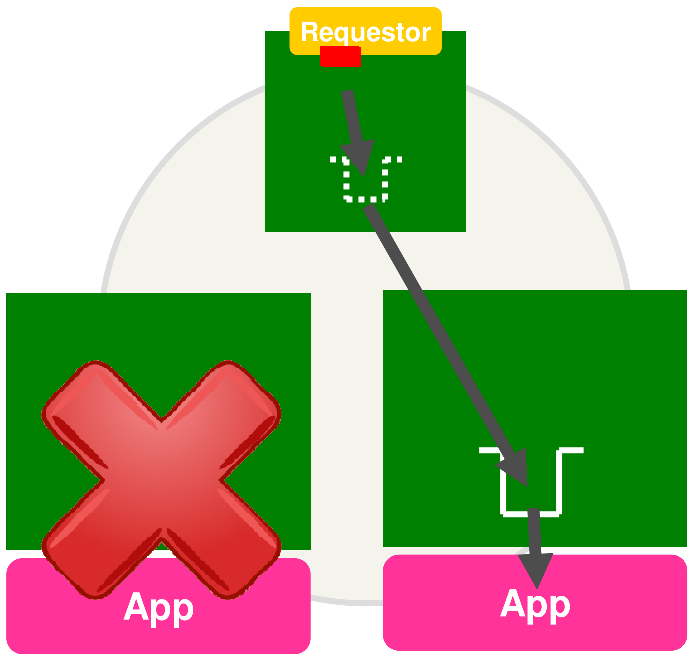
- When considering the load on the system from the proxy subscription traffic monitor the following queues.
  - The SYSTEM.INTER.QMGR.FANREQ queue on the subscriber queue manager.
  - The SYSTEM.INTER.QMGR.CONTROL queue on all other queue managers in the cluster.
- Any significant message backlog on these queues implies that either the rate of subscription change is too great for the system or a queue manager is not correctly functioning in the cluster.

- When PROXYSUB is set to FORCE on a WebSphere MQ V8 queue manager, the wildcarded proxy subscription covering all sub-topics will be generated and no subsequent proxy subscriptions will be generated.
- Prior to WebSphere MQ V8, setting PROXYSUB to FORCE for a topic will generate the wildcarded proxy subscription in advance but it will not stop the individual proxy subscriptions being created for each newly subscribed topic.
  - This can be achieved in a coarse grain way using the pscProxySubFlags tuning parameter, as documented in the info center.

Subscription availability

The continuously available point-to-point application

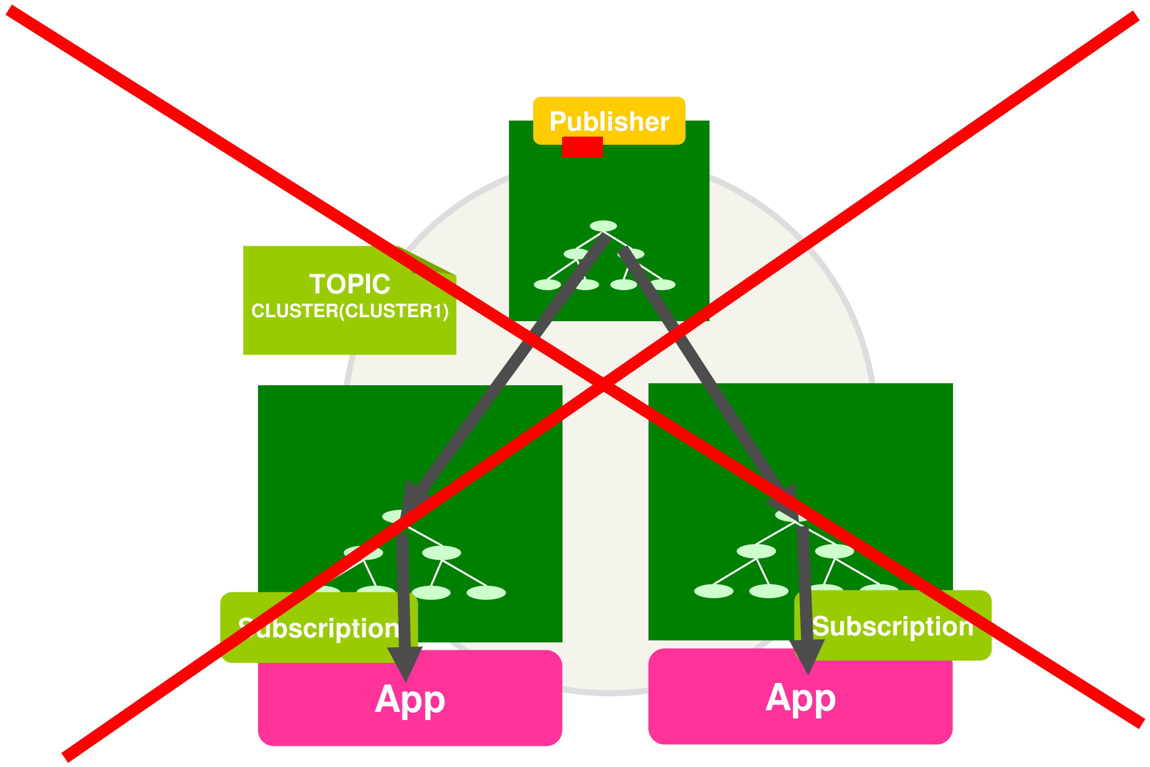
- When we need a point-to-point application to be continuously available we can cluster and duplicate its request queue.



- How do we get the same with a publish/subscribe application?....

The continuously available subscription

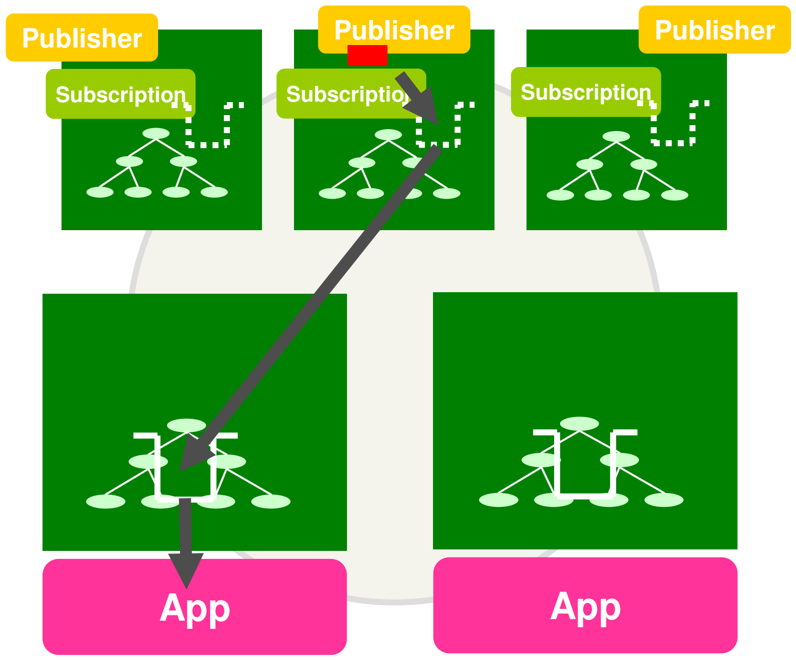
- The aim is to configure the system so that an application that processes publications can run multiple instances, across multiple queue managers.
- The instinct is to cluster the topic and duplicate the subscription.



- This is *not* the solution, it will lead to multiple copies of each publication

The continuously available subscription

- **Don't** define the topic as clustered *(this is **really** important).*
- Configure matching clustered queues, as for point-to-point and open these queues instead of subscribing.
- Define the same subscription on ***every*** queue manager where a publisher can connect.
  - Point those subscriptions at the **cluster queue**, leaving the target queue manager **blank.**



- Publish to the topic, and consume from the queues…

Hints and tips

Problem determination

- Double check your configuration.
- When in a cluster, check each queue manager’s cluster knowledge.
  - In V8 check the ***cluster state***
- When in a hierarchy, check your relationships.
- Check the error logs.
- Check your channels.
- Check your proxy subscriptions.
- Check your topic status.
- Watch for publication movement.
- Look for a build up of messages.

Problem determination 1/2

- Double check your configuration
  - Remember that topic objects inherit behaviour from higher topic nodes, this can affect how lower topics behave. Especially for things like SUBSCOPE and PUBSCOPE.
- When in a cluster, check each queue manager's cluster knowledge.
  - Use DISPLAY TOPIC(*) TYPE(CLUSTER)
  - From V8, check the **CLSTATE** to see if problems exist (see the info center for more details).
- When in a hierarchy, check your relationships.
  - Use DISPLAY PUBSUB
- Check the error logs.
  - It may be obvious but check the error logs of the queue managers with publishers and subscribers, and full repository queue managers, and any in between (topic hosts or hierarchy members).
- Check your channels.
  - If the channels to and from the queue managers are not running, publication traffic, proxy subscriptions, topic configuration (clusters) or relationships (hierarchies) will not flow.
- Check your proxy subscriptions on each queue manager.
  - Use DISPLAY SUB(*) SUBTYPE(PROXY)
- Check your topic *status.*
  - Use DISPLAY TPSTATUS( ) to show how a topic string will behave, this takes into account any administered topic objects that relate to it in the topic tree.

Problem determination 2/2

- Watch for publication movement.
  - Publications may be flowing, but not noticed.
  - Check the NUMMSGS value on the proxy subscriptions using DISPLAY SBSTATUS( ) to see if publications have been sent to the originator of the proxy subscription.
  - Check the MSGS value on the channels to see if they are flowing using DISPLAY CHSTATUS( )

- Look for a build up of messages.
  - In the event of problems, messages can build up on system queues. These may be because the system is producing messages too quickly or an error has occurred. Investigate to see if messages are still being processed, but slowly, or no messages are processed.
  - Queues which should be empty under normal running are:
    - SYSTEM.INTER.QMGR.PUBS (clusters)
      - Publications arriving from remote queue managers
    - SYSTEM.INTER.QMGR.FANREQ (clusters)
      - Requests to send proxy subscriptions (sometimes has one message, that's ok)
    - SYSTEM.INTER.QMGR.CONTROL (clusters)
      - Requests from other queue managers to register proxy subscriptions
    - SYSTEM.CLUSTER.TRANSMIT.QUEUE (or any other transmission queue involved)
      - All inter queue manager outbound messages
    - SYSTEM.BROKER.DEFAULT.STREAM (hierarchies)
      - Publications arriving from remote queue managers (and local 'queued' pub/sub applications)
    - SYSTEM.BROKER.CONTROL.QUEUE
      - Requests to register/deregister subscriptions
    - SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS (hierarchies)
      - Requests from other queue managers in the hierarchy.

Performance

- Distributing publishers and subscribers does cost in **latency**.
  - Keep high intensity publishers and subscribers together.
  - Or at least minimise their routes.
- But it does enable **horizontal scaling**.
  - Try to remove bottlenecks by having multiple subscription locations and routes to maximise throughput.
- Maximising **throughput**.
  - Message ordering is your enemy.
  - Multiple routes are possible at multiple levels:
    - Routing queue managers when using topic host routing.
      - Ideally for the same topics, but even across different topics can improve throughput.
    - Multiple cluster channels when using any clustering.
    - Multiple queue manager threads when processing inter-queue manager traffic.
- Don’t forget to consider the subscription and channel overheads.

Performance

- Check out the WebSphere MQ V7.5 Publish/Subscribe performance report.
  - SupportPac MP0C
  - (Found at http://www.ibm.com/support/docview.wss?uid=swg24033638 at the time of writing!)
  - Look out for V8 performance reports…

Summary

- The basics
- The possibilities
- Compare and contrast
- Scenarios

Questions & Answers

Legal Disclaimer