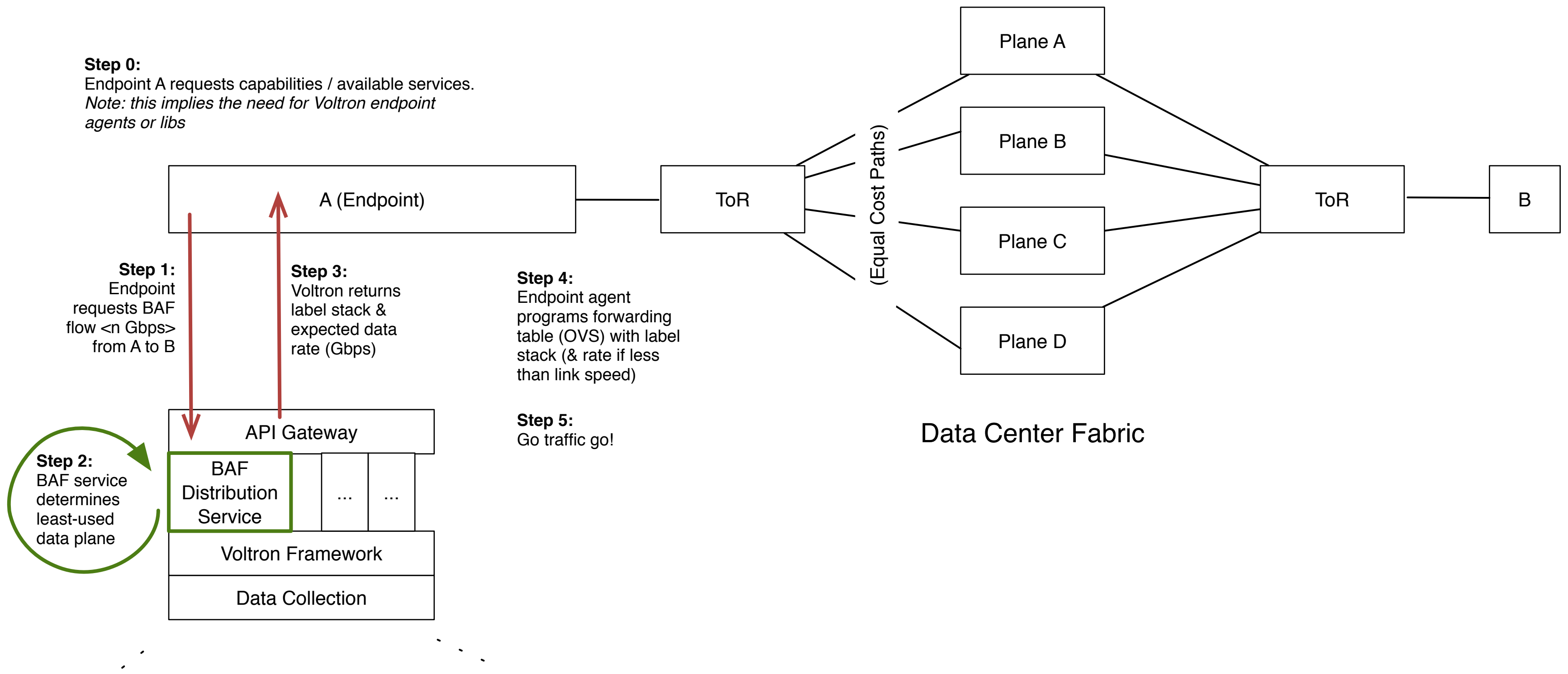
Use Case: BAF, originating at A and going to B

Note: we are not intending to suggest paths for everybody. Just for special types of flows.

Assumptions:
1) A knows how to get to Voltron
2) A knows its own connecting link speed (40 Gbps) and the data size of the BAF transfer (2TB)
3) A is a container running with Contiv networking
4) The API Gateway is a RESTful interface
5) The time between BAF service requests is greater than Volton's update interval
6) The DC is configured with sufficient MTU between A & B for the label stack



Step 0:
Endpoint A requests capabilities / available services.
Note: this implies the need for Voltron endpoint agents or libs

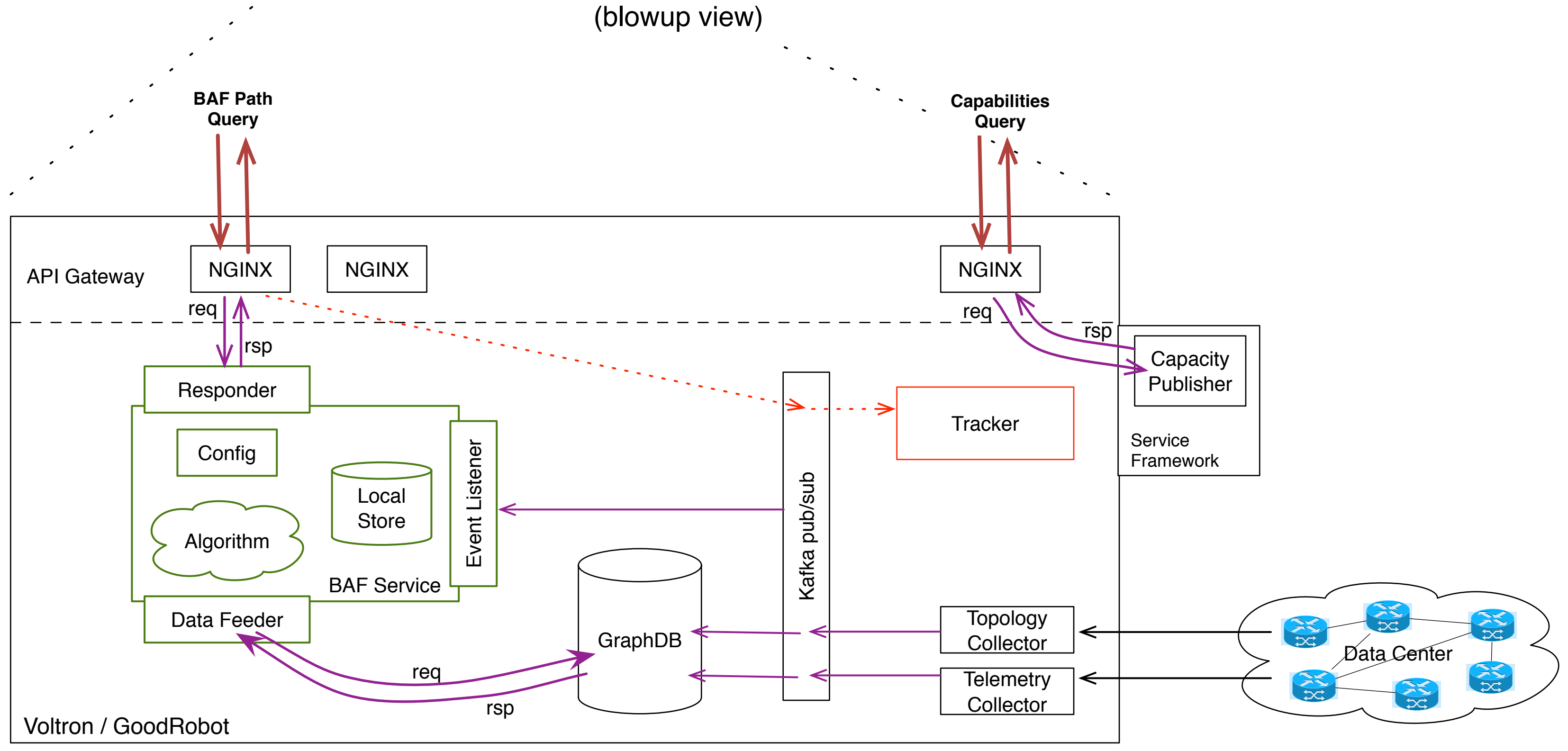
Step 1:
Endpoint requests BAF flow <n Gbps> from A to B

Step 2:
BAF service determines least-used data plane

Step 3:
Voltron returns label stack & expected data rate (Gbps)

Step 4:
Endpoint agent programs forwarding table (OVS) with label stack (& rate if less than link speed)

Step 5:
Go traffic go!



BAF Service Configuration Includes:
1) Definition of fabric planes & associated links
2) Initial service parameters:
- algorithm specification
- min time between GraphDB queries
- warmup time (before advertising API)
Note: This implies the need for a Service Lifecycle State Machine [loaded / warmup / active / inactive]

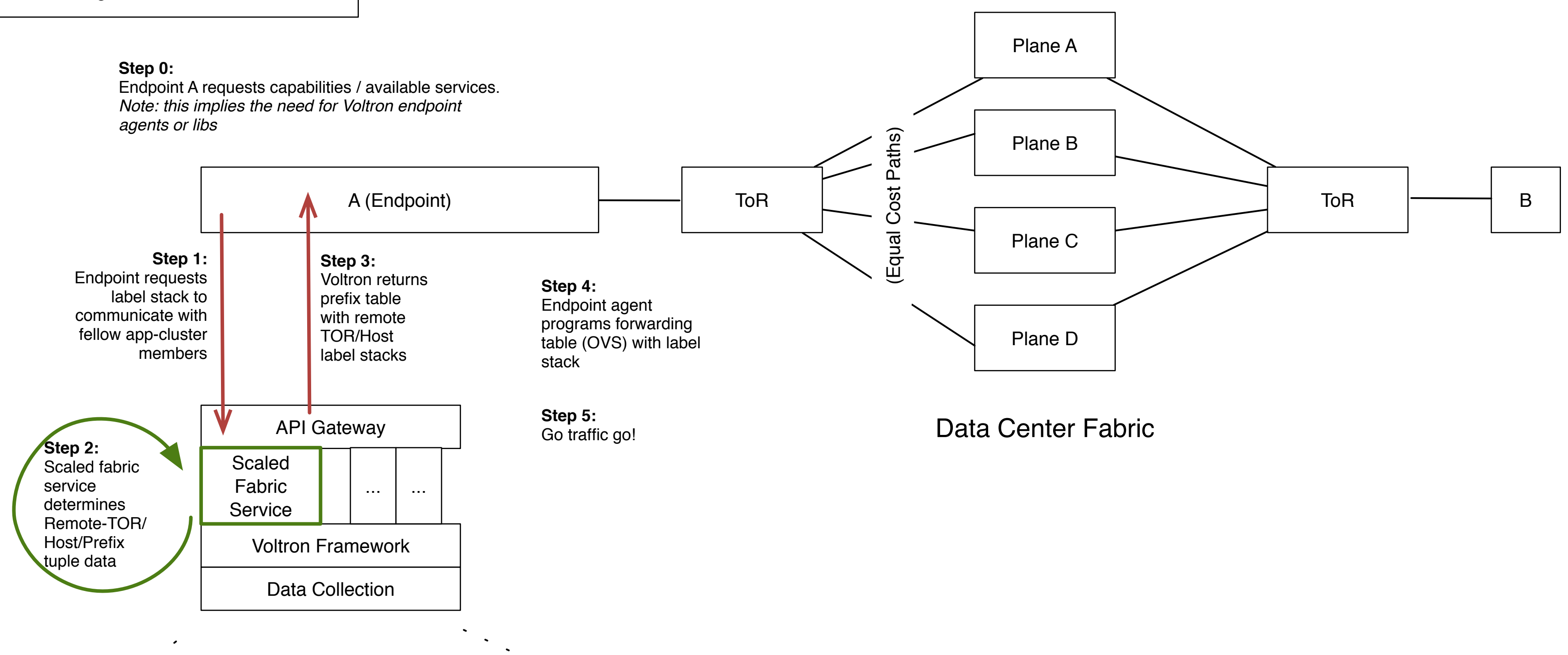
Situations to Consider
* How does BAF service deal with 100's to 1000's of endpoints? *one answer: have a set of best paths and round-robin through them*
* Due to telemetry data lag, need to be flexible of definition of "least used path"   i.e. a set of low utilization paths
* Need to consider synchronization between "best path set" at time t, and new information received from telemetry
* Due to high bi-sectional bandwidth in the fabric, the most probable point of bottle necks is flow polarization to a single egress leaf node (note Max Flow Capacity of paths).
* Can we streamline service needs and data acquisition? Perhaps Kafka topics is a solution

* Debug ability - what events & data do we need to log and/or track within Voltron?

Use Case: Scaling the Fabric

Note: this use case describes a model wherein endpoint prefix scale in the containerized/virtualized data center exceeds forwarding table scale in the DC fabric itself. Initial case is single-tenant.

Assumptions:
1) A knows how to get to Voltron
2) A is a container running with Contiv networking
3) A is a member of an application cluster and therefore cannot be NAT'd behind a host IP
4) A has an IPv4/v6 address that is not summarized at its local TOR
5) TOR prefix-SID is within the SRGB
6) Host prefix-SID is outside of the SRGB
7) The API Gateway is a RESTful interface
8) The DC is configured with sufficient MTU between A & B for the label stack



Step 0:
Endpoint A requests capabilities / available services.
Note: this implies the need for Voltron endpoint agents or libs

Step 1:
Endpoint requests label stack to communicate with fellow app-cluster members

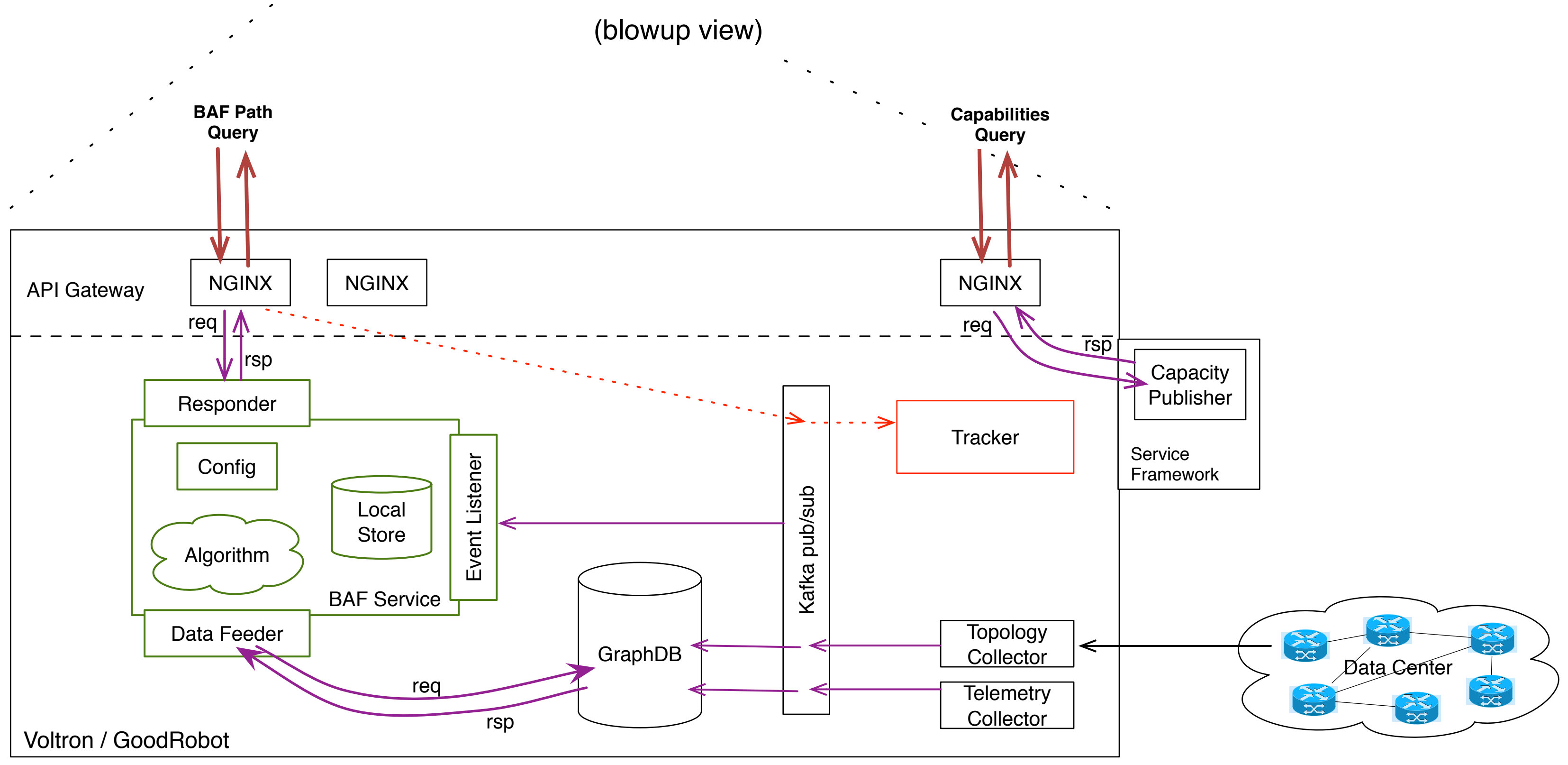
Step 2:
Scaled fabric service determines Remote-TOR/ Host/Prefix tuple data

Step 3:
Voltron returns prefix table with remote TOR/Host label stacks

Step 4:
Endpoint agent programs forwarding table (OVS) with label stack

Step 5:
Go traffic go!

Fabric Scale Service Configuration Includes:
1) Harvesting TOR prefix-SID and Host label data
  - Host label may simply be an EPE label for outgoing TOR interface
2) Harvesting Host/container prefix data
3) Initial service parameters:
- creation of tuples from TOR-SID, Host-SID, prefix
- min time between GraphDB queries
- warmup time (before advertising API)
Note: This implies the need for a Service Lifecycle State Machine
[loaded / warmup / active / inactive]



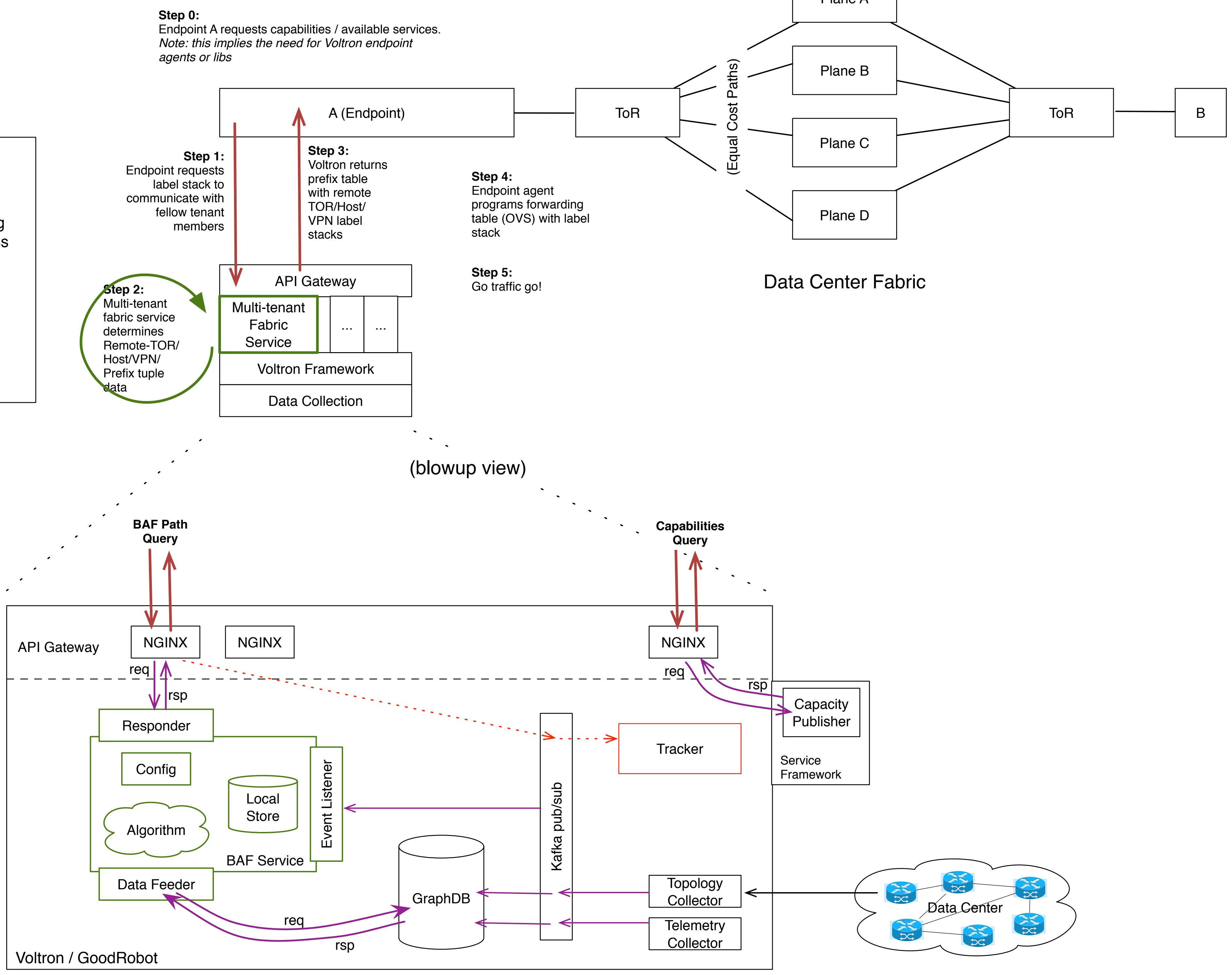
Situations to Consider
* How does BAF service deal with 100's to 1000's of endpoints? *one answer: have a set of best paths and round-robin through them*
* Due to telemetry data lag, need to be flexible of definition of "least used path"   i.e. a set of low utilization paths
* Need to consider synchronization between "best path set" at time t, and new information received from telemetry
* Due to high bi-sectional bandwidth in the fabric, the most probable point of bottle necks is flow polarization to a single egress leaf node (note Max Flow Capacity of paths).
* Can we streamline service needs and data acquisition? Perhaps Kafka topics is a solution

* Debug ability - what events & data do we need to log and/or track within Voltron?

Use Case: Multi-Tenant Fabric

Note: this use case describes a multi-tenant containerized/virtualized data center wherein endpoints may be mobile within the DC and endpoints with different tenant IDs might have overlapping IPv4/v6 addresses

Assumptions:
1) A knows how to get to Voltron
2) A is a container running with Contiv networking
3) Voltron is able to associate A's IPv4/v6 address with A's tenant_ID
5) TOR prefix-SID is within the SRGB
6) Host prefix-SID is outside of the SRGB
7) The API Gateway is a RESTful interface
8) The DC is configured with sufficient MTU between A & B for the label stack



Step 0:
Endpoint A requests capabilities / available services.
Note: this implies the need for Voltron endpoint agents or libs

Step 1:
Endpoint requests label stack to communicate with fellow tenant members

Step 2:
Multi-tenant fabric service determines Remote-TOR/ Host/VPN/ Prefix tuple data

Step 3:
Voltron returns prefix table with remote TOR/Host/ VPN label stacks

Step 4:
Endpoint agent programs forwarding table (OVS) with label stack

Step 5:
Go traffic go!

Fabric Scale Service Configuration Includes:
1) Harvesting TOR prefix-SID and Host label data
  - Host label may simply be an EPE label for outgoing TOR interface
2) Harvesting Container VPN/tenant, and prefix data
3) Initial service parameters:
- creation of tuples from TOR-SID, Host-SID, VPN-SID, prefix
- min time between GraphDB queries
- warmup time (before advertising API)
Note: This implies the need for a Service Lifecycle State Machine
[loaded / warmup / active / inactive]

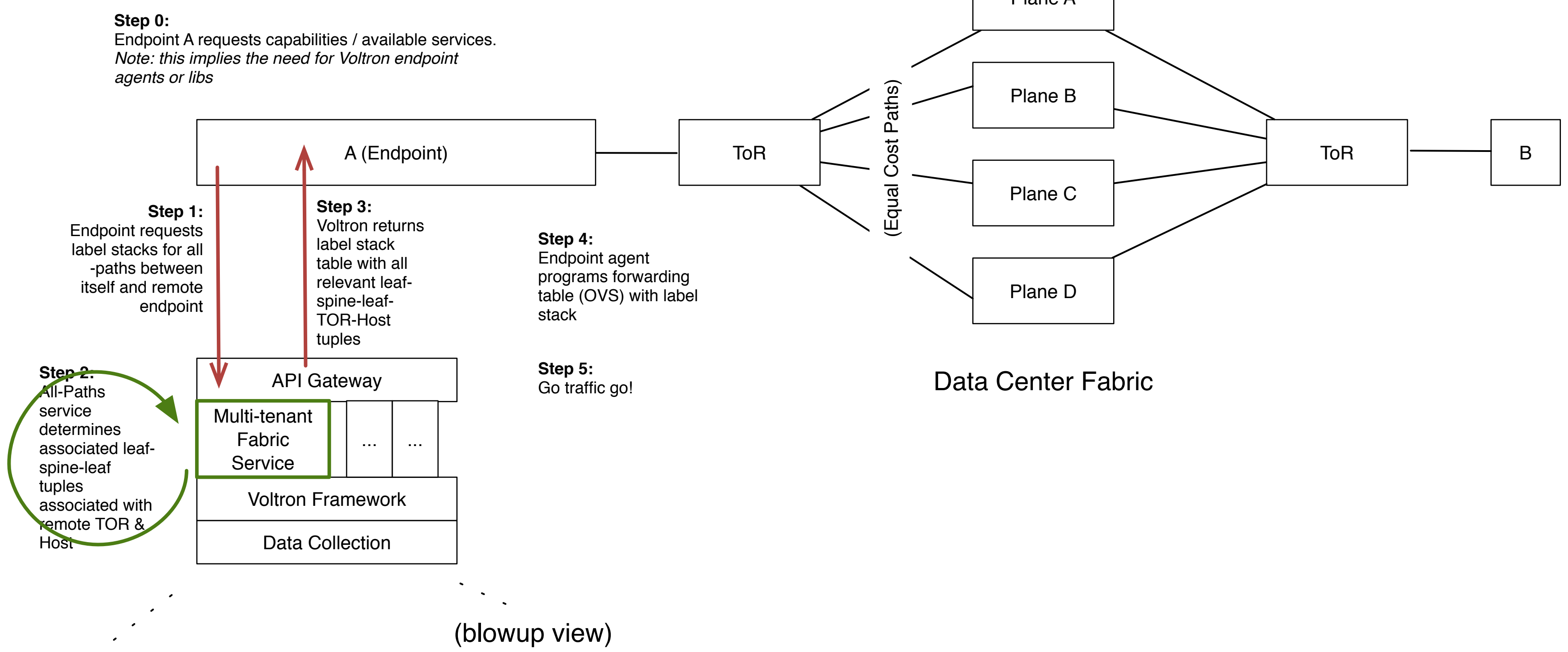
Situations to Consider
* How does BAF service deal with 100's to 1000's of endpoints? *one answer: have a set of best paths and round-robin through them*
* Due to telemetry data lag, need to be flexible of definition of "least used path"   i.e. a set of low utilization paths
* Need to consider synchronization between "best path set" at time t, and new information received from telemetry
* Due to high bi-sectional bandwidth in the fabric, the most probable point of bottle necks is flow polarization to a single egress leaf node (note Max Flow Capacity of paths).
* Can we streamline service needs and data acquisition? Perhaps Kafka topics is a solution

* Debug ability - what events & data do we need to log and/or track within Voltron?

Use Case: All-Paths Service

Note: this use case describes a scenario where an OAM tool needs to test the liveliness (and perhaps latency) of all links in the fabric between itself and another host/endpoint. The micro service returns all-paths between the two endpoints in response to the request.

Assumptions:
1) A knows how to get to Voltron
2) A is a container running with Contiv networking
3) All fabric nodes (Spine, leaf, TOR) have prefix-SIDs within the SRGB
4) Host prefix-SID is outside of the SRGB
7) The API Gateway is a RESTful interface
8) The DC is configured with sufficient MTU between A & B for the label stack



Step 0:
Endpoint A requests capabilities / available services.
Note: this implies the need for Voltron endpoint agents or libs

Step 1:
Endpoint requests label stacks for all -paths between itself and remote endpoint

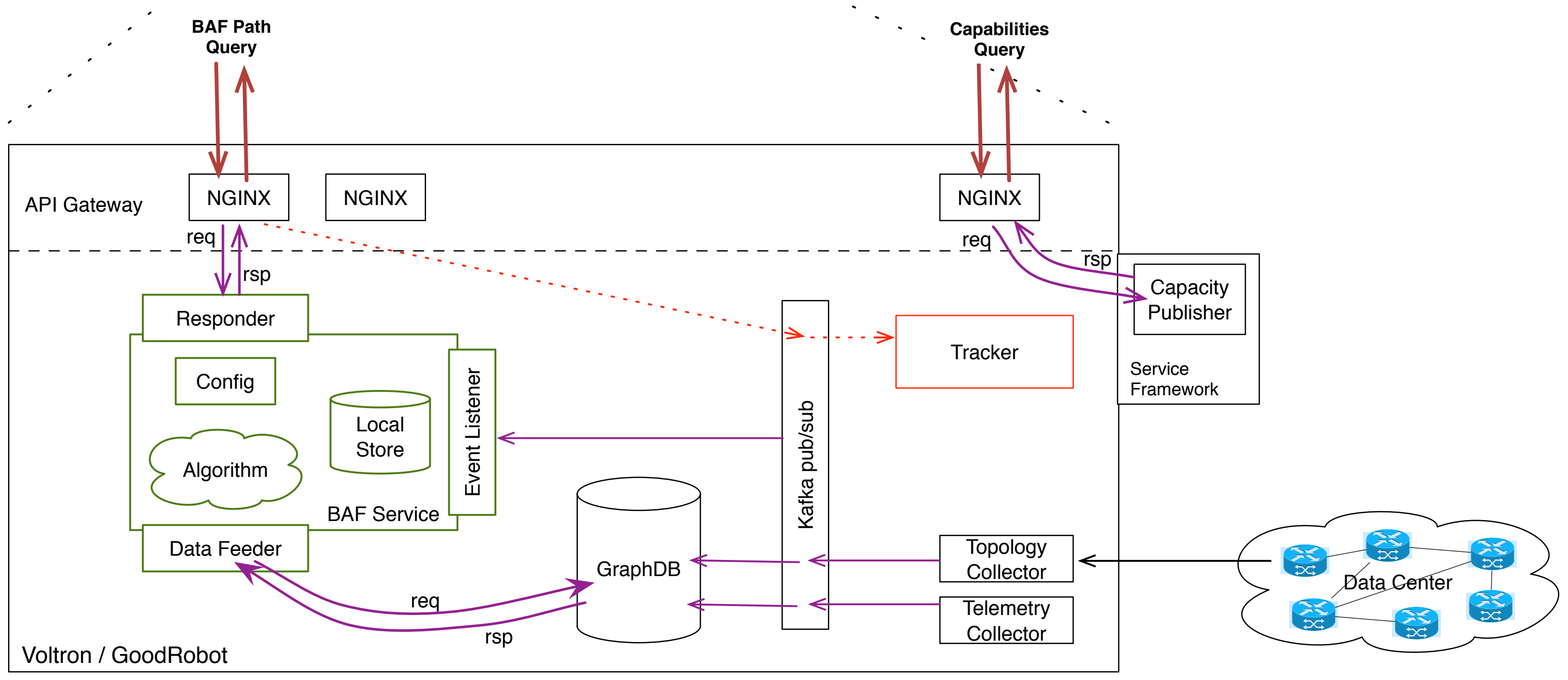
Step 2:
All-Paths service determines associated leaf-spine-leaf tuples associated with remote TOR & Host

Step 3:
Voltron returns label stack table with all relevant leaf-spine-leaf-TOR-Host tuples

Step 4:
Endpoint agent programs forwarding table (OVS) with label stack

Step 5:
Go traffic go!

All-Paths Service Configuration Includes:
1) Harvesting TOR prefix-SID and Host label data
  - Host label may simply be an EPE label for outgoing TOR interface
2) Harvesting of fabric node (spine and leaf) prefix-SID (and possibly adj-SID) data
3) Initial service parameters:
- creation of tuples from leaf-spine-leaf to a given remote TOR/Host
- example table size: assuming 3-stage CLOS built across 4-planes, TOR uplinks to 4 leafs (1 per plane), and leaf uplinks to 48 spines within its plane. Once traffic reaches a given leaf it has already entered a DC plane, therefore it has 48 paths to spine, then all spines have a single path to the appropriate egress leaf, so the table size is (4*48*1) = 192 L-S-L tuples or paths.
- warmup time (before advertising API)
Note: This implies the need for a Service Lifecycle State Machine
[loaded / warmup / active / inactive]

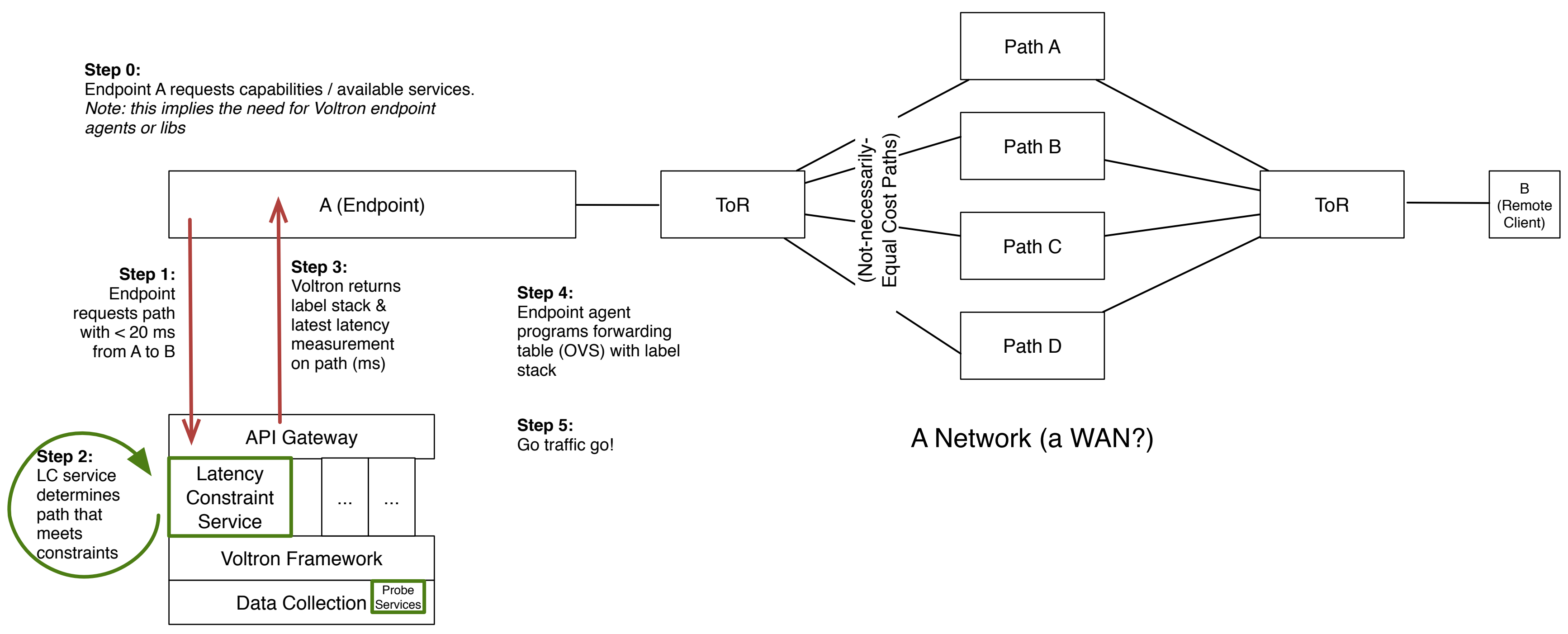


Situations to Consider
* How does BAF service deal with 100's to 1000's of endpoints? *one answer: have a set of best paths and round-robin through them*
* Due to telemetry data lag, need to be flexible of definition of "least used path" i.e. a set of low utilization paths
* Need to consider synchronization between "best path set" at time t, and new information received from telemetry
* Due to high bi-sectional bandwidth in the fabric, the most probable point of bottle necks is flow polarization to a single egress leaf node (note Max Flow Capacity of paths).
* Can we streamline service needs and data acquisition? Perhaps Kafka topics is a solution

* Debug ability - what events & data do we need to log and/or track within Voltron?

Use Case: Low Latency Flow (sub 20ms), originating at A and going to B

Assumptions:
1) A knows how to get to Voltron
2) A knows its own connecting flow rate (1 Gbps) and its peak latency tolerance (20ms)
3) A is a container running with Contiv networking
4) The API Gateway is a RESTful interface
5) The time between LC service requests is greater than Volton's update interval (required?)
6) The DC is configured with sufficient MTU between A & B for the label stack



Step 0:
Endpoint A requests capabilities / available services.
Note: this implies the need for Voltron endpoint agents or libs

Step 1:
Endpoint requests path with < 20 ms from A to B

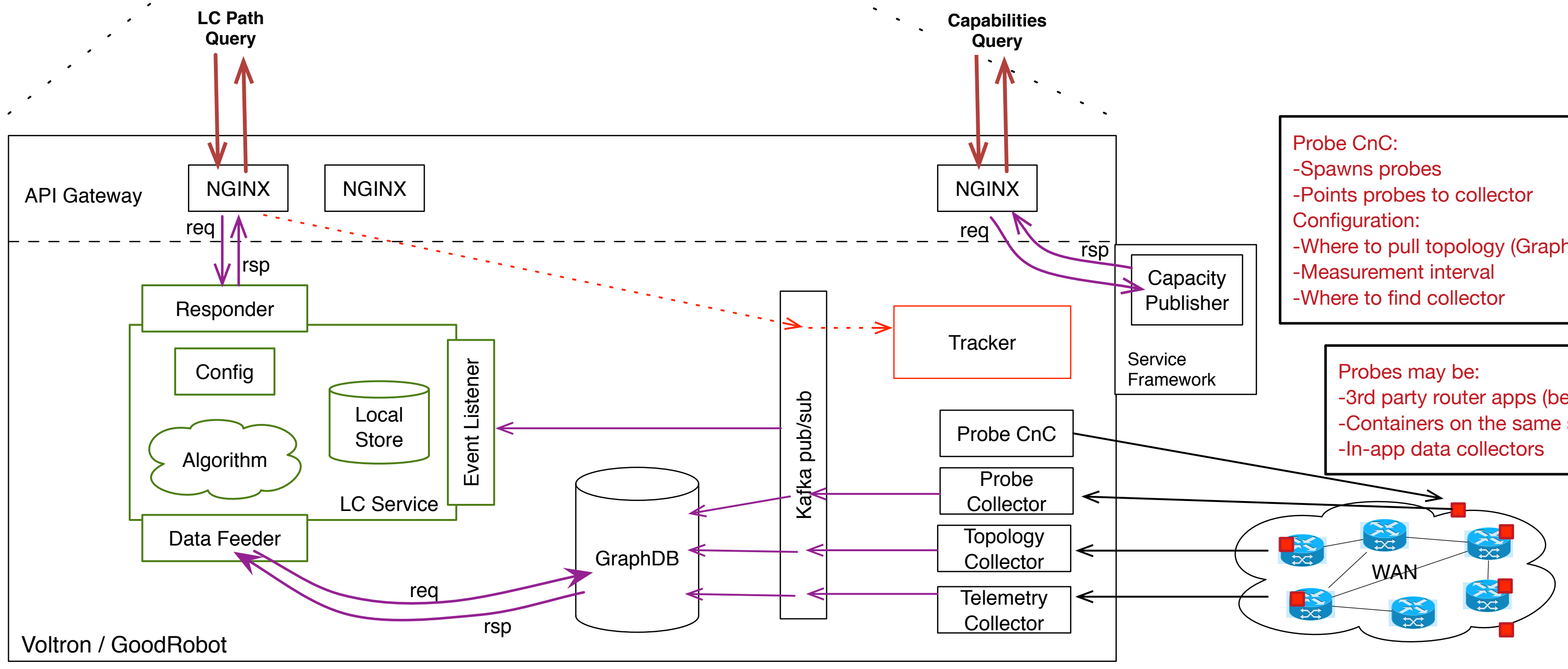
Step 2:
LC service determines path that meets constraints

Step 3:
Voltron returns label stack & latest latency measurement on path (ms)

Step 4:
Endpoint agent programs forwarding table (OVS) with label stack

Step 5:
Go traffic go!

(blowup view)



Probe CnC:
-Spawns probes
-Points probes to collector
Configuration:
-Where to pull topology (GraphDB)
-Measurement interval
-Where to find collector

Probes may be:
-3rd party router apps (best option?)
-Containers on the same servers
-In-app data collectors

Latency Constraint Service Configuration Includes:
1) Location of liveness check for LC data publisher
2) Initial service parameters:
- min time between GraphDB queries
- % of paths covered by measurements before advertising API
- minimum freshness of latency data

Situations to Consider
* How does BAF service deal with 100's to 1000's of endpoints? *one answer: have a set of best paths and round-robin through them*
* Due to telemetry data lag, need to be flexible of definition of "least used path"   i.e. a set of low utilization paths
* Need to consider synchronization between "best path set" at time t, and new information received from telemetry
* Due to high bi-sectional bandwidth in the fabric, the most probable point of bottle necks is flow polarization to a single egress leaf node (note Max Flow Capacity of paths).
* Can we streamline service needs and data acquisition? Perhaps Kafka topics is a solution

* Debug ability - what events & data do we need to log and/or track within Voltron?