M365 Messaging Substrate Transport [NEW]

By Chelsea Gichohi, Emanuel Yilma and Anish Chaudhuri

Introduction:

Microsoft Exchange Transport processes billions of messages daily, ensuring that emails are delivered promptly and efficiently. On-call engineers (OCEs) are responsible for monitoring and resolving these issues within the Exchange transport system.

OCEs receive alerts corresponding to different Severity Levels, with Sev-3 being the least severe (being read by OCEs in the form of an email, and Sev-2 and Sev-1 being more severe (being read by OCEs in the form of a text or phone call, with the expectation that they are acting upon the error within 15 minutes). Currently, if a singular message in a MailboxServer is expired, OCEs will receive a Sev-3 alert.

Problem:

While Sev-3 alerts notify OCEs when a singular message is expired, no Sev-2 or Sev-1 alert exists to notify OCEs during a critical or intolerable uptick of expired messages across all Exchange Servers. As such, lack of a Sev-2 or Sev-1 alert thus enables widespread failures of the Exchange to go unnoticed. Moreover, the migration of Exchange towards allowing for tenants to set their own expiration times on messages poses a challenge regarding diagnostics. Existing Geneva metrics that display the number of messages stuck in the pipeline for greater than 24 hours on dashboards are no longer emblematic of Microsoft's ability to deliver messages on behalf of their customers. As such, existing metrics and alerting systems need to be replaced and deprecated ASAP.

"Priority 1" issues also include lack of automation regarding tenant, network and call stack information via On Call Advisor, and "Priority 2" issues include prediction and correlation detection between 'past known issues' and 'present behavior.' In addition, 'Priority 1' issues include the lack of a larger-scale stuck messaging alerting system & dashboard for active stuck messages aggregated within short-term intervals (e.g., 30 minutes).

Proposed Solution:

The proposed solution introduces a ***metric***, namely the total, aggregated number of expired messages (where 'expired' messages are messages that are in the Exchange Transport pipeline past their custom expiration time), a ***dashboard*** to display the metric akin to the *SubmissionQueueTransport dashboard*, as well as an ***alerting system*** that provides a ***Sev-2 alert*** to OCEs when the number of expired messages is beyond a threshold of 100 messages.

In addition, we seek to provide OCEs with clear visualization and step-by-step guides on how to solve these issues. We will do this by relying on ***RCA*** (Root-Cause-Analysis) frameworks, updating the Wiki

Guide, and ensuring that automation for ***OnCallAdvisor*** includes Call Stack info, Network Diagnostics, and Tenant information.

In Summary:

- Detect the number of Stuck Messages in Exchange Transport Pipeline that are past their custom expiry time (Expired Messages).
- Create a dashboard displaying this number and implement a more efficient alerting system when threshold of 100 is exceeded.
- Detect the number of Active Stuck Messages in Exchange Transport Pipeline that are stuck for ~30 minutes (e.g., 15-45 minutes). Do not count CFM (control-flow message: process-to-process communications; e.g., sending a like).
- Create a dashboard displaying this number and implement a more efficient alerting system when threshold of 1000 is exceeded [normal priority]
- Represent key metrics (top forest, top server, etc.) via graphs below each of these dashboards.
- Automate OCA tools and write Wiki guidelines for OCEs to monitor and resolve issues.

Dashboard & Alert Feature Proposals:

Dashboards & Metrics*:*

Priority	Item	Notes
0	Replicate Submission Queue Dashboard. We are replicating the graphs in the dashboard, but the time aggregation is different.	Review Documentation, Alerting System, Gain Familiarity With Geneva Monitoring.
0	Create New Metric that Aggregates Count of Total Messages Stuck Per Forest (internal to Geneva)	We need to figure out how to aggregate count of messages internal within Geneva (likely using SUM)
0	Create Alerting System for Above Alert that throws a Sev-2 alert for OCEs.	100 messages 'stuck past their expiration date' or for more than 24 hours should be our threshold for **a Sev-2 Alert.** **[threshold confirmed by Shaun]**
0	Incorporate Information Regarding Message Expiration Time (will be working with things beyond Geneva).	Will likely interact with RDAP. Instead of only considering messages to be 'stuck' based on whether they have been in the pipeline for over 24 hours, consider messages to also be 'stuck' when they have been in the pipeline past their 'expiration time.'
1	Create Dashboard Representing Short-Term Message Aggregation (Higher Count of Messages Stuck for a Shorter Amount of Time)	
1	Create Alerting System for Above Alert that throws a Sev-2 alert for OCEs.	>1000 messages stuck for ~30 minutes should be our threshold for **a Sev-2 Alert.**

		[threshold yet to be determined, Shaun should get ~4 Sev-2 alerts]

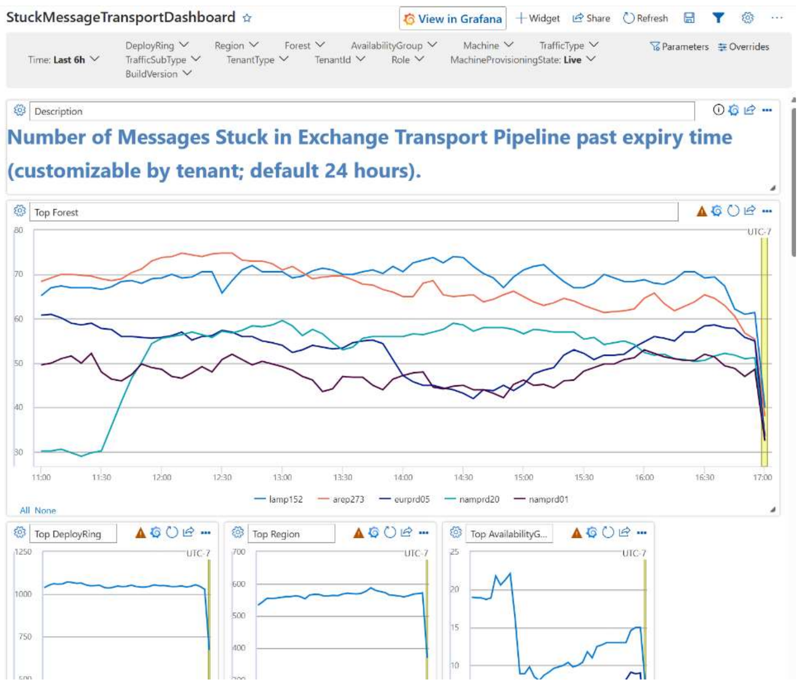
Troubleshooting/RCA with On Call Advisor:

Priority	Item	Notes
1	Be able to access expiration time of respective messages via Powershell. This will be used to define the new metric. ***A simpler way to do this is by updating cmdlet Get-Message to also provide expiration time, instead of creating a new cmdlet.***	Will require RDAP, and creation of a new metric.
2	Figure out relevant tenants via Powershell and display in list in OnCallAdvisor.	Utilize Get-Queue-Diversity, along with commands Sanjeev showed
1	Figure out call stack information and insert in OnCallAdvisor Report.	Use Get-ThreadStackGrouping and insert in OCA
1	Figure out Network Diagnostics and insert in OnCallAdvisorReport, especially for short-term **SEV-2 alert where ~1000 messages for > 30 minutes**.	NET VIEW??
1	Edit Guide for OCEs	N/A

Similarity Feature (P2):

Priority	Item	Notes
2	Notifications with links to other resolved events or issues that mimic behavior of current event.	Include all rows in OnCallAdvisor.
2	Algorithm that figures out whether a past event simulates a present event. An elementary approach to this is by looking at each 'relevant graph' and figuring out whether, for each value, the difference between past value and present value is within 1% of the max amplitude. ***<u>If so, insert past 'known issue' in OnCallAdvisor.</u>***	Before applying this algorithm, we need to ensure that the information and graphs that we look at are variable and it is extremely unlikely for values to approximate one another. Pick the minimum 'value' of this algorithm as the most likely candidate.
3	Graph of probabilities of type of events or past issues that have been already categorized.	Run the above algorithm for all types of past resolved issues and rank. Display on graph.

Example:



Number of Messages Stuck in Exchange Transport Pipeline past expiry time (customizable by tenant; default 24 hours).

Appendix:

The Exchange Transport Pipeline has three parts. The Submission phase is where emails are submitted to the Exchange transport system. The Delivery phase handles the routing of the email to its destination. The Hub serves as the central point in the Exchange system, where various agents process messages before delivery. However, a small percentage of messages can become stuck within the Exchange system, causing emails to fail to progress through the Exchange Transport System and remain undelivered. Stuck messages, which is defined as messages that are stuck in the service, can occur due to several reasons; system errors, configuration problems, or agent failures within the hub. Agents are responsible for ensuring that only appropriate messages are filtered out using anti-spam, anti-phishing, content inspect, and routing. Occasionally, faulty logic within agents results in problems that can cause traffic to get stuck, resulting in a backlog of undelivered messages.

Refer to Get-Queue Divesity and commands like $d = get-databaseAvailabilityGroup | where {$_.NormalTrackTransportEnabled} $d -> forest and member servers $queues = Get-Queue –Server BYSPR00M0674 $queues[-3] -> NextHopDomain, Status {Ready, Retry, Ready} Get-Queue-Diversity –Forest [categorizing by organizations, senders, recipients] -> aggregating by forest $m = Get-Message –Queue BYSPR00MB0674 $m[0] -> expirationTime *For specific project –dag NMPRD... $servers = Get-CentralAdminMachine –Filter {DAG –eq 'NAMPRD...'} -showAll*

.

- Cross 2000 messages at 30-minutes eventually crosses the 1hour-1day messages Why don't we alert

What specific metrics needs to be published
What data do we need to aggregate and How do we want to aggregate (how frequently is it happening? Every 5 min? whats happening behind the scenes? And how do want to present the data?

M365 Messaging Substrate Transport [ORIGINAL]

By Chelsea Gichohi, Emanuel Yilma and Anish Chaudhuri

1. Background and Goal

Background:

Exchange transport processes billions messages daily, ensuring that emails are delivered promptly and efficiently. The Exchange transport has three parts. The Submission phase is where emails are submitted to the Exchange transport system. The Delivery phase handles the routing of the email to its destination. The Hub serves as the central point in the Exchange system, where various agents process messages before delivery. However, a small percentage of messages can become stuck within the Exchange system, causing emails to fail to progress through the Exchange Transport System and remain undelivered. Stuck messages can occur due to several reasons, such as system errors, configuration problems, or agent failures within the hub. Agents are responsible for processing messages, performing tasks like spam filtering, content inspection, and routing. Occasionally, faulty logic within agents results in problems that can cause traffic to get stuck, resulting in a backlog of undelivered messages. On-call engineers are responsible for monitoring and resolving these issues within the Exchange transport system. They utilize ICM to track, manage, and resolve incidents related to stuck messages and other issues. While the current dashboards provide basic monitoring capabilities, they lack comprehensive visualization and alerting for stuck messages. The proposed solution aims to address this issue by detecting stuck messages, aggregating them within a forest (a collection of servers and systems that are connected and work together as a unified network, allowing multiple servers to share settings, data, directories), and generating alerts if their number exceeds a specified threshold. Providing engineers with clear visualization through Geneva dashboards will enable them to quickly identify and address these issues.

Objectives:

- Detection the number of Stuck Messages within a determined threshold
- Aggregate stuck messages within forests to understand scale of issue
- Implement a more efficient alerting system when threshold is exceeded

- Utilize Geneva dashboards to visualize key metrics (top forest, top server, etc.)
- Develop tools and guidelines for on-call engineers to monitor and resolve issues

Plan:

For MS 365 Transport Exchange, we are developing a new dashboard for on-call engineers (OCEs). This dashboard will monitor the volume of stuck messages and include graphs with relevant metrics to assist OCEs in identifying and resolving issues efficiently. Our project aims to detect stuck messages within the Exchange transport system, aggregate them in a forest, and trigger alerts if the number of such messages exceeds a certain threshold. The data will be visualized in Geneva dashboards, providing essential information such as top forests, top servers, and the total number of stuck messages. By producing actionable optics and alerts, we aim to reduce the customer impact of stuck messages and minimize troubleshooting time for OCEs.

1. Create two separate alerts for “stuck messages within a short duration” and “stuck messages within a long duration” for OCE customers. ***Numerical amount is aggregated from the various forests.*** These two alerts, with their respective dashboards will show OCEs with relevant information, including:
    - Network Information and Connectivity for each module/package *for ‘stuck messages within a short duration’*. This will output a list of packages that are not connected.
    - Agent Information – Will output a list of agents that have completed their tasks, as well as before that, a list of agents that are to be run by each email; hopefully, this will point to which agent’s logic is stuck.
    - Top Machine, Top Forest graphs for both dashboards – if top machine or top forest indicates that a *small amount* < *n* machines or forests are impacted, these will be listed.
2. Integration of *cmdlets* via PowerShell Scripts for all alerts
    - Tenant & mailbox information and forest information will be outputted.
3. Similarity between graphs and dashboards for all alerts -> raising issues
    - Will analyze issues resolved by OCEs in the past, and see if relevant graph information ‘approximates’ graph information from the past; if so, will direct OCE to earlier issue resolved in order to highlight similarity between past issue and present issue (and thus potential similarity of solutions)

3.1 A dashboard corresponding to probability of type *k* issue.

- By using past diagnostic information, and similarity between that information, will output a graph corresponding to the probability of a certain type of issue (types of issues are defined by similarity of documentation relating to issues resolved after resolution).

<u>Timeline:</u>

Weeks	Project Milestone Description	Category*	Deliverable
1-3	• Ramping up working with Exchange / Substrate		PM and Dev Spec

	• Understanding probe, monitor, alert framework using Geneva • Understand what stuck messages are and how they can be detected, current LAM workflow • PM and Dev Spec		
4-6	• Working on publishing data to Geneva • Aggregating data and monitoring • Dashboards		Dashboard for stuck messages
Checkpoint		Mid-Point Check In	
7-9	• Laying out plan for investigation of those alerts • Validate the alerting framework using test data • Using OncallAdvisor for quick action on alerting		Sev-2 alert TSG OCA
10-11	• Working on Final Demo and Presentation		
Final		Final Presentation	

Dashboard & Alert Feature Proposals:

Classifications of Two Alerts *(by message volume and time):*

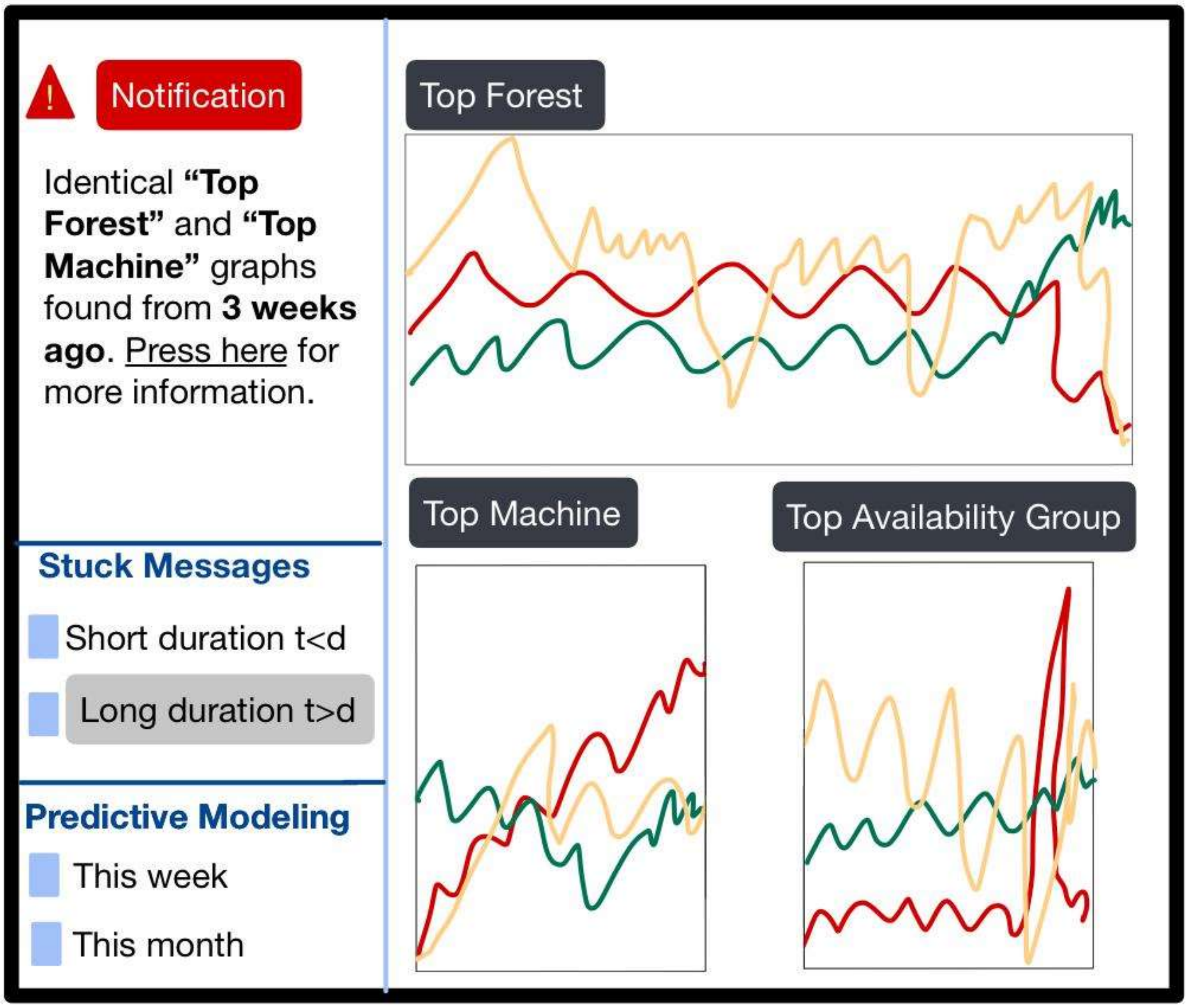
Priority	Item	Notes
0	Short Duration Stuck Messages [Time Threshold & Message Threshold]	We need to look at the amount of messages that we'd consider a necessary Sev-2 Alert, as well as figure out how to aggregate count of messages by forest.
0	Long Duration Stuck Messages [Time Threshold & Message Threshold]	We need to look at the amount of messages that we'd consider a necessary Sev-2 Alert, as well as figure out how to aggregate count of messages by forest.
0	Aggregate Network Connectivity For Each Package (highlight the packages/modules that are **not connected)**	Assumes that this is not a hardware/code issue and likely an issue with the third-party services that Microsoft uses. (*within short messages)*
0	Aggregate Network Connectivity For Each Forest (highlight the packages/modules that are **not connected)**	I think this should be displayed top-most. (*within short messages)*
0	Network Connectivity For Each Machine (highlight the machines that are **not connected)**	In the case of a hardware issue, this information is relevant.

		(*within short messages)*
0	Agent List [agents that need to be run]	(*for long durations)*
0	Agents Completed [agents that do not need to be run]	(*for long durations)*
1	Timer on Each Agent [*before alert is sent?*]	(*for long durations*)

Similarity Feature:

Priority	**Item**	**Notes**
1	Notifications with links to other resolved events or issues that mimic behavior of current event.	
1	Algorithm that figures out whether a past event simulates a present event. An elementary approach to this is by looking at each 'relevant graph' and figuring out whether, for each value, the difference between past value and present value is within 1% of the max amplitude.	Before applying this algorithm, we need to ensure that the information and graphs that we look at are variable and it is extremely unlikely for values to approximate one another. Pick the minimum 'value' of this algorithm as the most likely candidate.
2	Graph of probabilities of type of events or past issues that have been already categorized.	Run the above algorithm for all types of past resolved issues and rank. Display on graph.

Example Dashboard Model with Proposed Features:



PowerShell Scripts (*refer to recording with Sanjeev and figure out the pertinent command information)*

Priority	Item	Notes
0	Be able to access expiration time of respective messages via Powershell.	[some relevant commands here] $d = get-databaseAvailabilityGroup \| where {$_.NormalTrackTransportEnabled} $d -> forest and member servers $queues = Get-Queue –Server BYSPR00M0674 $queues[-3] -> NextHopDomain, Status {Ready, Retry, Ready} Get-Queue-Diversity –Forest [categorizing by organizations, senders, recipients] -> aggregating by forest

		$m = Get-Message –Queue BYSPR00MB0674 $m[0] -> expirationTime *For specific project –dag NMPRD...* *$servers = Get-CentralAdminMachine –Filter {DAG –eq 'NAMPRD...'} -showAll*
0	Figure out relevant tenants via Powershell and display in list under Geneva Dashboard.	-
0	Cluster messages that have been stuck in a while by forest name.	

Items to further clarify

- What data exists that documents the similarity between
- How can we make the search less costly
- How can we improve efficiency in identifying similar patterns?
- What is a similar graph?
- How can we utilize historical data to identify reoccurring bugs?
- How can we utilize historical data to predict future bugs?
- For our intern project plan, can you explain on call advising for priority one?