Overload Control for Scaling WeChat Microservices

Authors - Hao Zhou, Ming Chen, Qian Lin, Yong Wang, Xiaobin She, Sifan Liu, Rui Gu, Beng Chin Ooi, Junfeng Yang

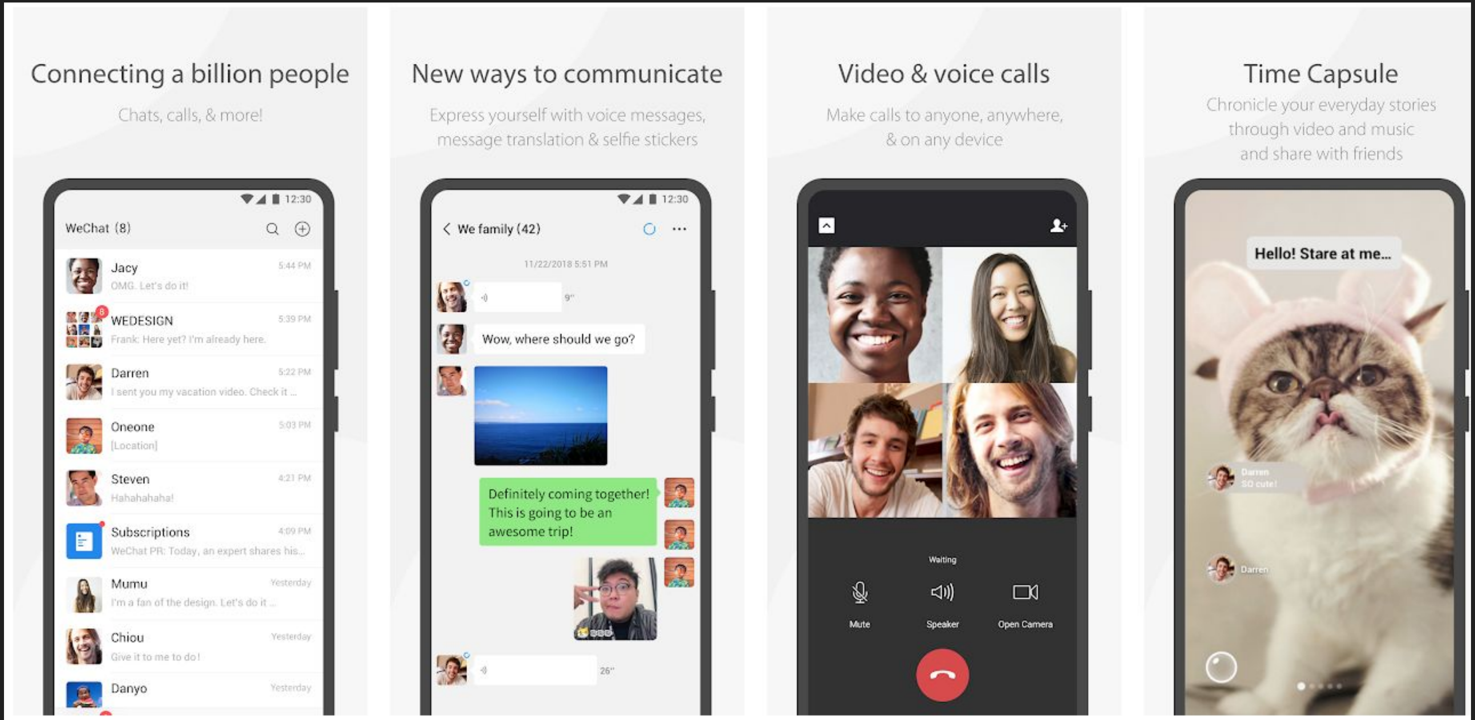
Presented by - Kirtan

Overview

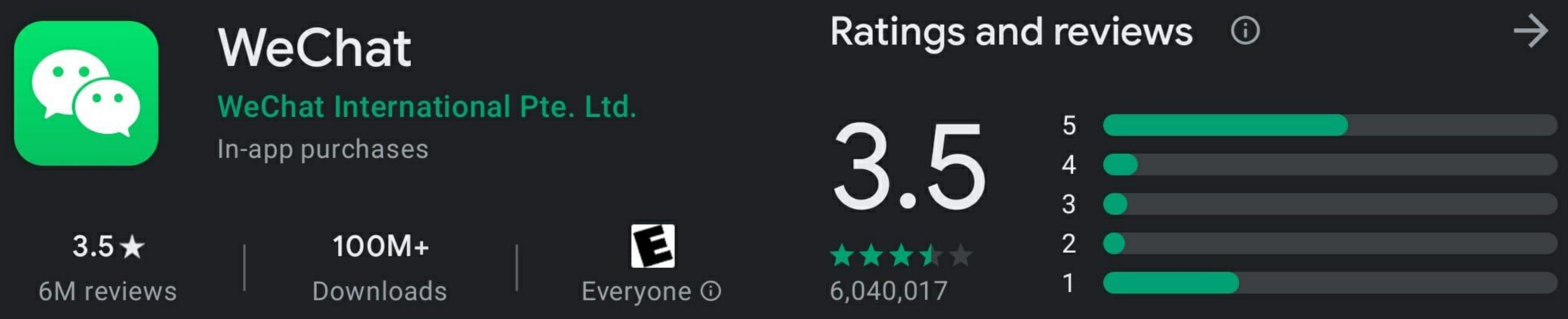
1. What is WeChat?
2. Introduction.
3. Background.
4. Overload in WeChat.
5. DAGOR.
6. Evaluation.
7. Conclusion.

Slides Count 26

What is WeChat?



What is WeChat?



Introduction - Overload in Application.

1. Overload control aims to mitigate service irresponsiveness.
2. Limited Computing resources affordability.
3. Complex Microservice Architecture are difficult to manage for overload.
    a. Microservices must be monitored.
    b. Microservices should not handle overload independently.
    c. Adapt to the service changes.

Introduction - DAGOR

Overload control scheme, called DAGOR, for a large-scale, account-oriented microservice architecture.

1. Service Agnostic.
2. Adaptive with respect to service changes.
3. Practical solution of overload control for an operational microservice system.

Background - Service Architecture

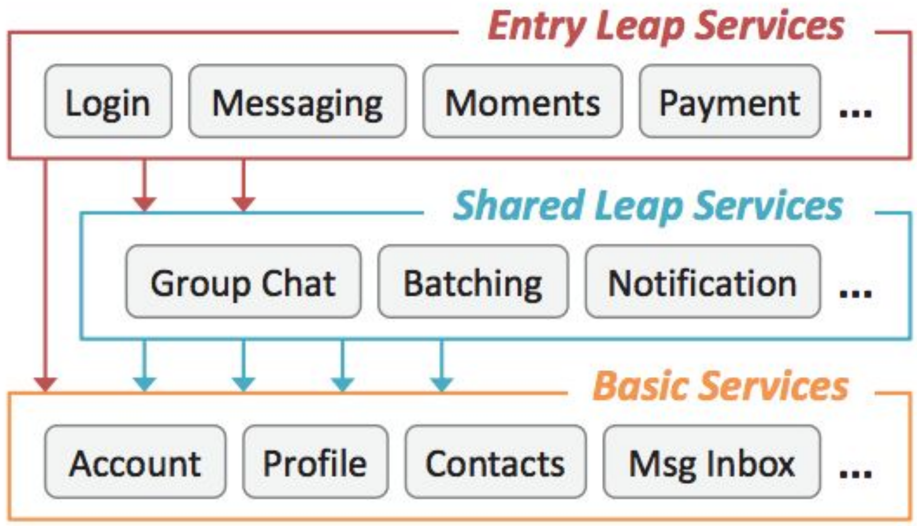
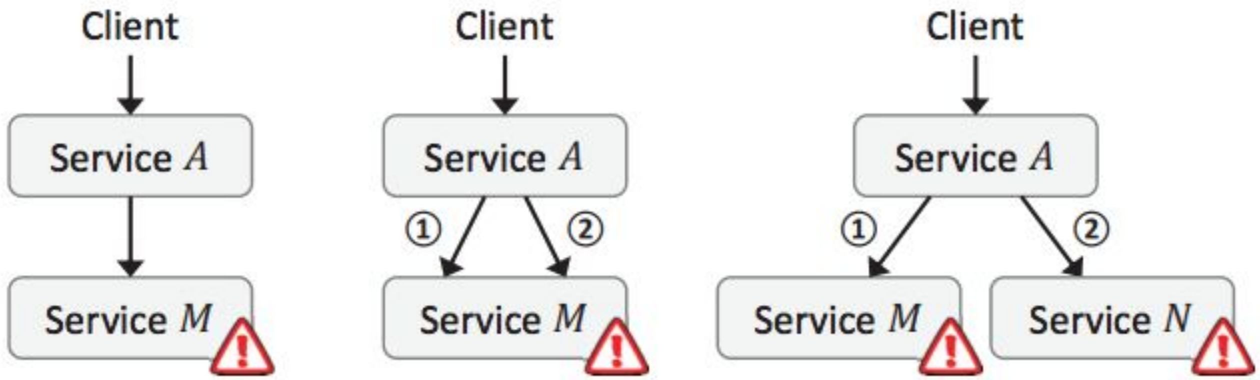


Figure 1: WeChat's microservice architecture.

Background - Service Architecture

1. 3000+ services.
2. Over 10^{10} requests per day to Entry Service. (10 times of the daily average during the Chinese Lunar New Year)
3. 20,000+ machines.
4. 1000+ changes made in a day.

Background - Subsequent Overload



(a) Form 1 (b) Form 2 (c) Form 3

Figure 2: Common overload scenarios.

Background - Subsequent Overload

For random load shedding hampers the overall system performance.

If A service calls M service. *(M sheds 50% load randomly)*

Success rate of A for 1 request is 0.5C, where C is request count.

Success rate of two Subsequent request becomes 0.25C.

So the probability of successful service request reduces with more number of microservices involved in the subsequent request.

DAGOR Overload Control

1. Service Agnostic.
   a. Applicable to all kinds of services.
   b. Not rely on any service-specific information
   c. Unaffected by improper configuration of service.
2. Independent but Collaborative.
   a. Run on the granule of individual machine.
   b. Collaboration between different machines.
3. Efficient and Fair.
   a. Computational resources (i.e., CPU and I/O) wasted on the failed service tasks are minimized.

DAGOR - Overload Detection

1. **Average waiting time** of requests in the pending queue.
2. **Request Processing time** isn’t local.
3. High **CPU utilization** doesn’t always mean overload.

Window based monitoring system.

1. Refreshes every **2000** requests. (calculated 347222 requests per second, 173 refreshes per second across entry services)
2. **500ms** request timeout.
3. **20ms** average threshold for overload.

DAGOR - Admission Control

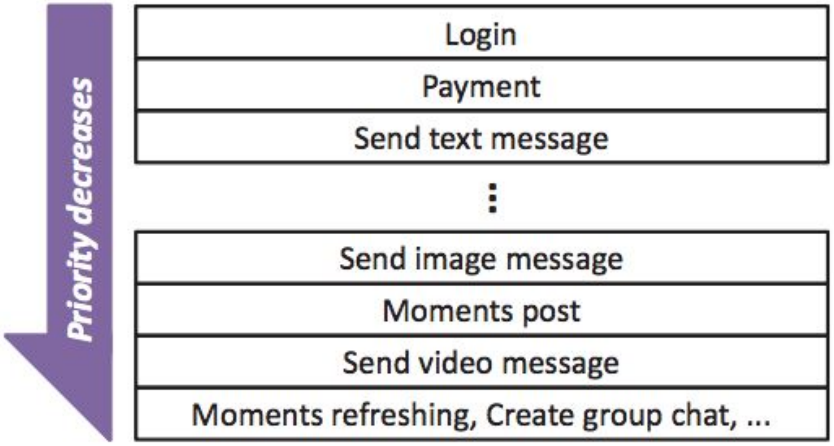


Figure 3: Hash table storing the business priorities of actions to perform in the WeChat entry services.

DAGOR - Admission Control

1. Business-oriented Admission Control
    a. Hashmap for business priorities (AID:Priority), Login, Money Transfer etc.
    b. Priorities are recursively passed to subsequent services.
    c. Rarely changes over time.
2. User-oriented Admission Control.
    a. Provides more fine grained priority range if used along with Business Priority.
    b. Avoids partial discarding of requests in Business Priority.
    c. User Priority is dynamically generated.
3. Session-oriented Admission Control.
4. Adaptive Admission Control.
    a. Finds the right **(B, U)** priority setting on the fly.
5. Collaborative Admission Control.
    a. Piggyback updated admission control settings.
    b. Do a local check before sending the request.

DAGOR - Admission Control

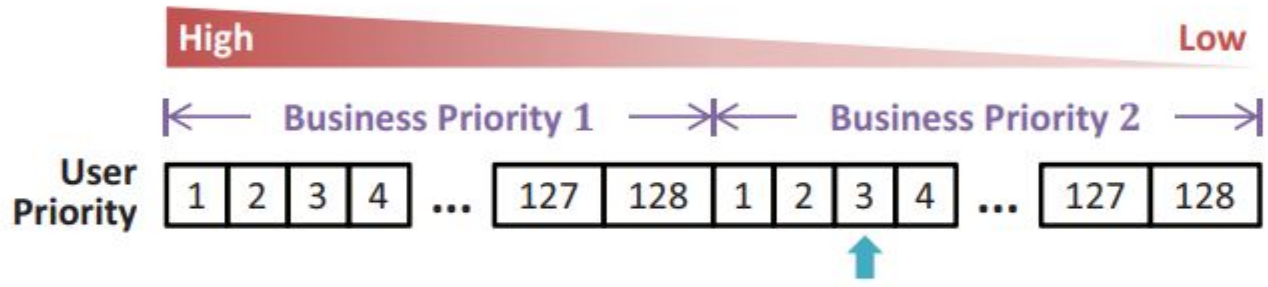


Figure 4: The compound admission level.

DAGOR - Adaptive Admission Control

Global: max. admission level of business priority $\mathcal{B}_{\max}$
Global: max. admission level of user priority $\mathcal{U}_{\max}$
Global: admitted request counters $C[\mathcal{B}_{\max}][\mathcal{U}_{\max}]$
Global: incoming request counter N

DAGOR - Adaptive Admission Control

Procedure ResetHistogram():

$N \leftarrow 0$

foreach $c \in C$ **do** $c \leftarrow 0$

Input: service request r

Procedure UpdateHistogram(r):

$N \leftarrow N + 1$

if r is admitted **then** $C[r.\mathcal{B}][r.\mathcal{U}] \leftarrow C[r.\mathcal{B}][r.\mathcal{U}] + 1$

DAGOR - Adaptive Admission Control

α - 5% drop for overload.

β - 1% increase.

Input: boolean flag f_{overload} indicating overload
Output: compound admission level
Procedure `CalculateAdmissionLevel`(f_{overload}):

- $N_{\text{exp}} \leftarrow N$
- **if** $f_{\text{overload}} = \mathit{true}$ **then** $N_{\text{exp}} \leftarrow (1 - \alpha) \cdot N_{\text{exp}}$
- **else** $N_{\text{exp}} \leftarrow (1 + \beta) \cdot N_{\text{exp}}$
- $(\mathcal{B}^*, \mathcal{U}^*) \leftarrow (0, 0)$
- $N_{\text{prefix}} \leftarrow 0$
- **for** $\mathcal{B} \leftarrow 1$ **to** $\mathcal{B}_{\max}$ **do**
  - **for** $\mathcal{U} \leftarrow 1$ **to** $\mathcal{U}_{\max}$ **do**
    - $N_{\text{prefix}} \leftarrow N_{\text{prefix}} + C[\mathcal{B}][\mathcal{U}]$
    - **if** $N_{\text{prefix}} > N_{\text{exp}}$ **then return** $(\mathcal{B}^*, \mathcal{U}^*)$
    - **else** $(\mathcal{B}^*, \mathcal{U}^*) \leftarrow (\mathcal{B}, \mathcal{U})$
- **return** $(\mathcal{B}^*, \mathcal{U}^*)$

DAGOR - Workflow

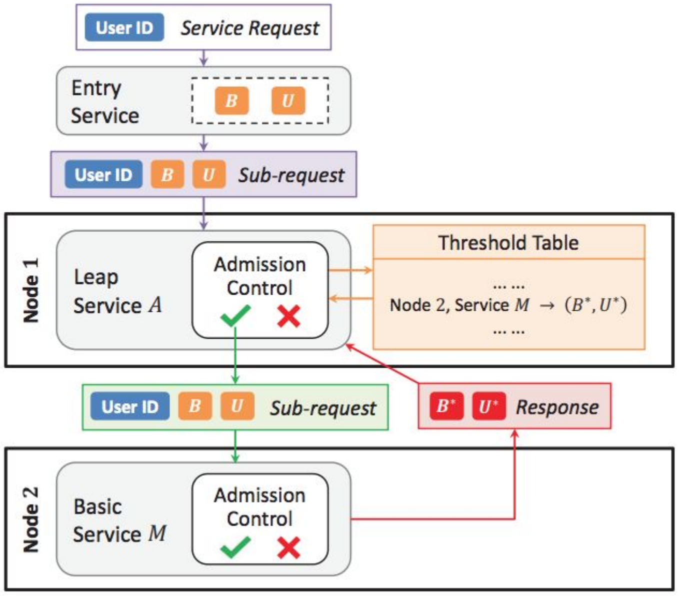


Figure 5: Workflow of DAGOR overload control.

Evaluation

1. Encryption service (Mi) and Messaging service (Ai).
2. In-house cluster. (over 3 machines for each service)
3. Each machine has Intel Xeon E5-2698 @ 2.3 GHz CPU and 64 GB DDR3 memory.
4. 10 Gigabit Ethernet.

Evaluation - Two workloads

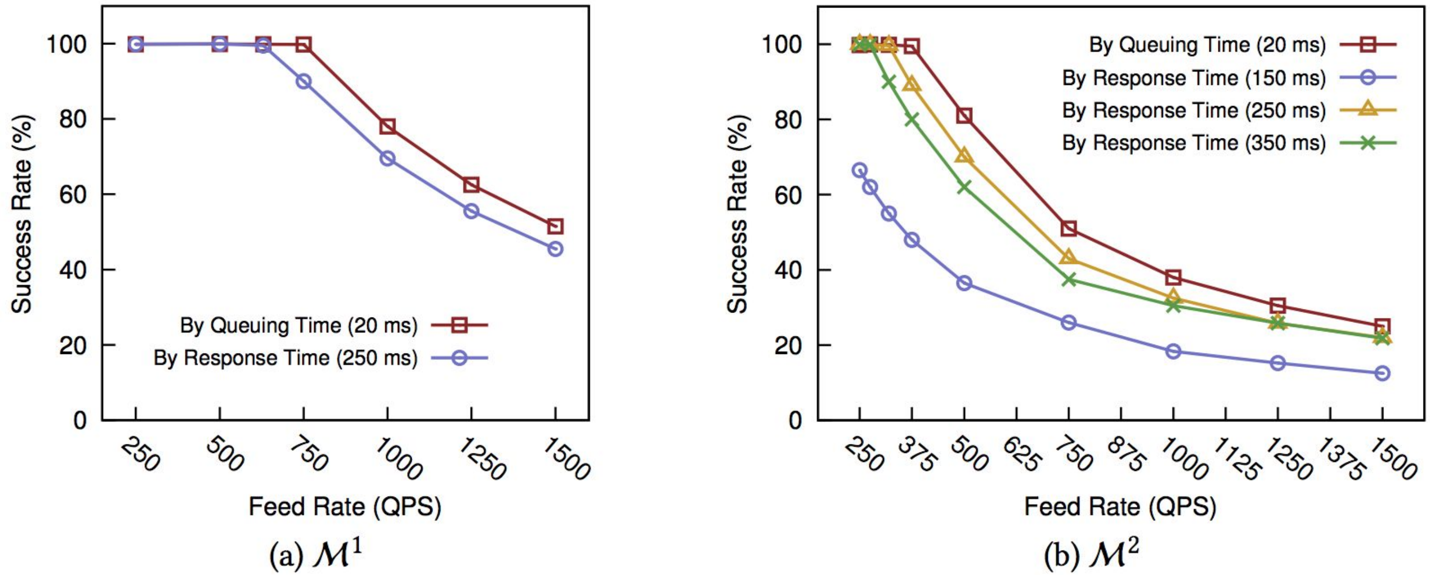
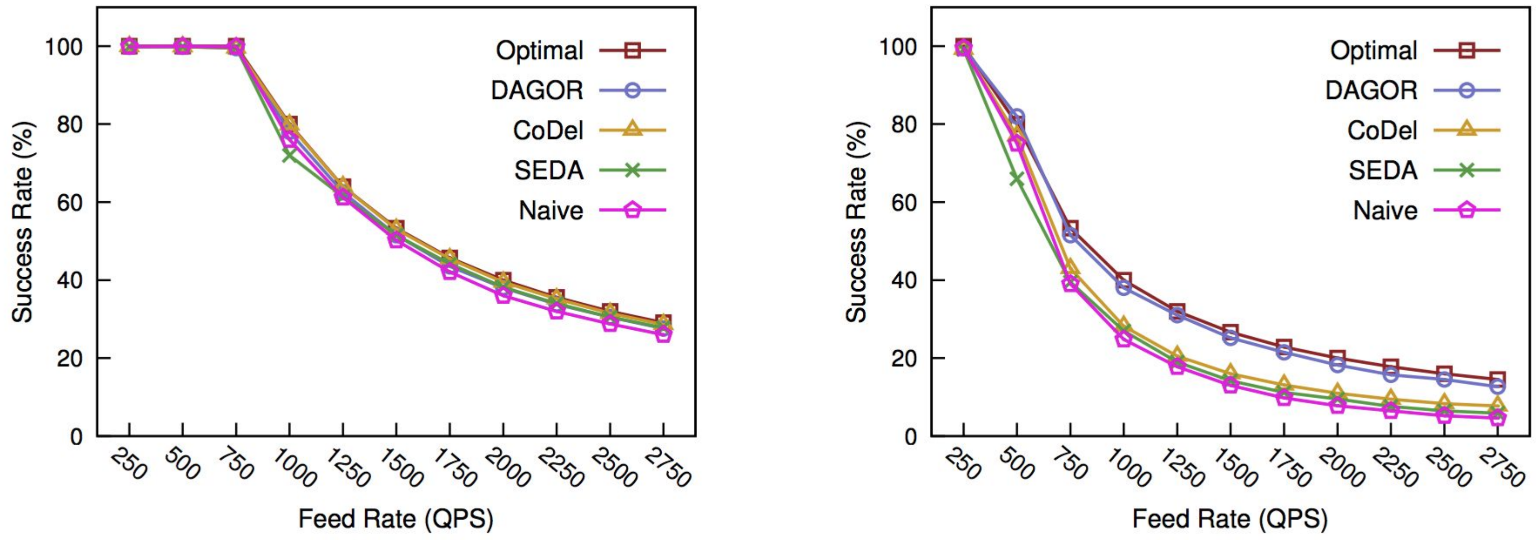


Figure 6: Overload detection by different indicators of load profiling: queuing time vs. response time.

Evaluation - Comparing with other Overload mechanisms



(a) $\mathcal{M}^1$

(b) $\mathcal{M}^2$

Figure 7: Overload control with increasing workload.

Evaluation - QPS fixed at 1500

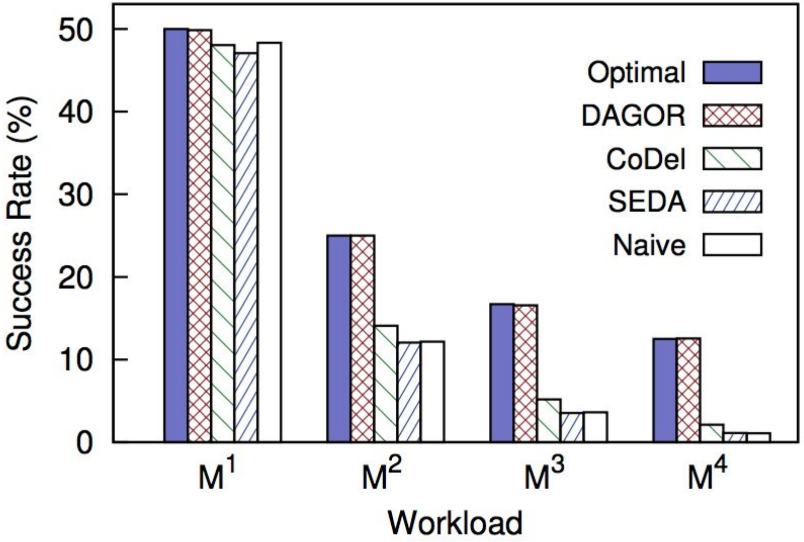


Figure 8: Overload control with different types of workload.

Evaluation - Subsequent overload

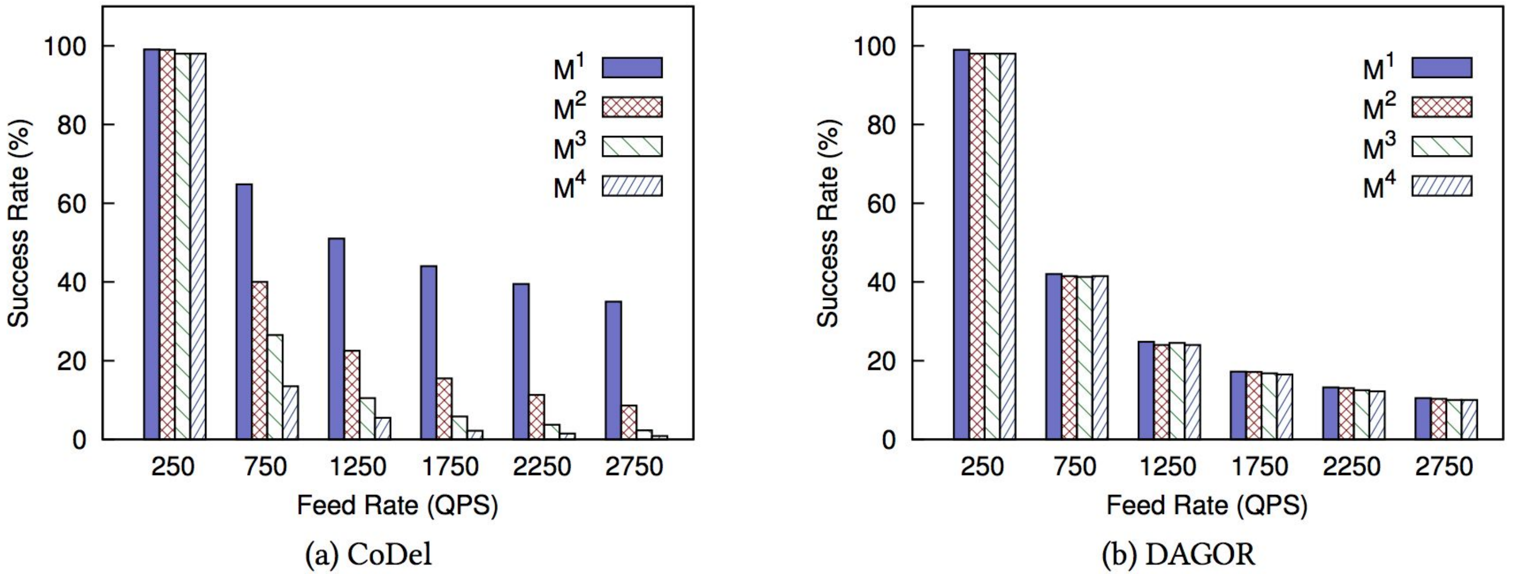


Figure 9: Fairness of overload control.

Conclusion

DAGOR works well with subsequent overload!

Thank You.