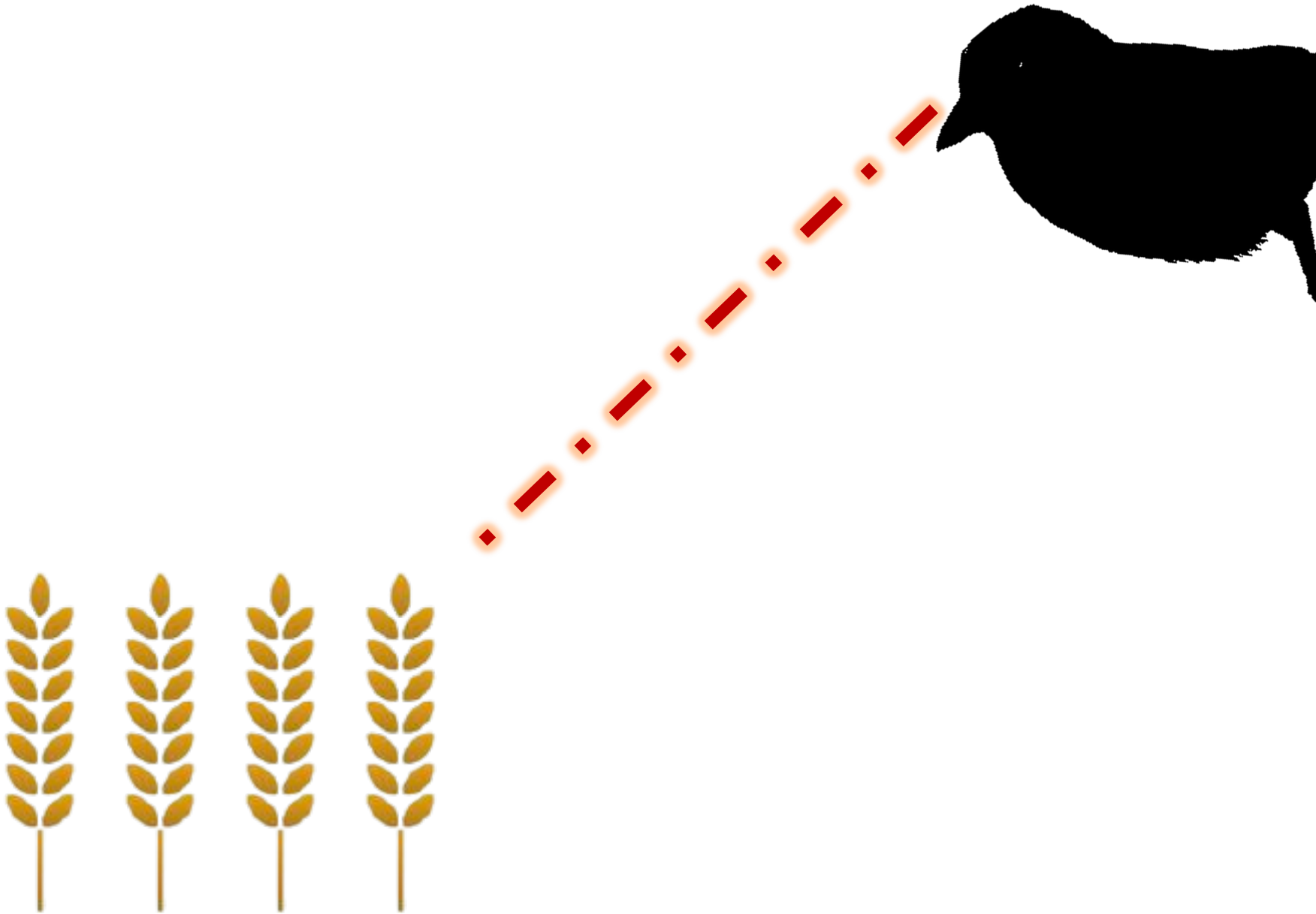
Kafka

@RoboNovotny

UINF/KOPR jeseň 2022



Čo je Kafka?

- Apache Kafka
- pôvodne v LinkedIn na zber clickstreamov

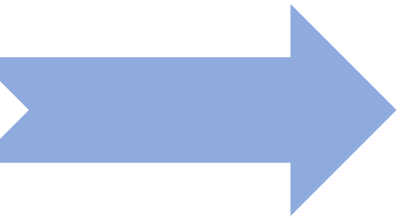
Čo je Kafka?

distribuovaný commit log

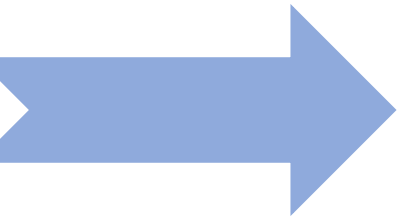
Čo je Kafka?

distributed
streaming platform

Producenti a konzumenti



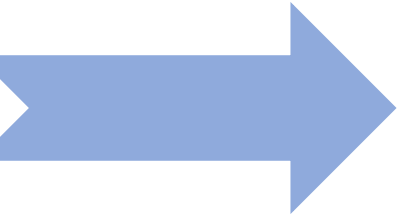
Správy a témy | messages and topics



- kľúč (voliteľný)
- dáta

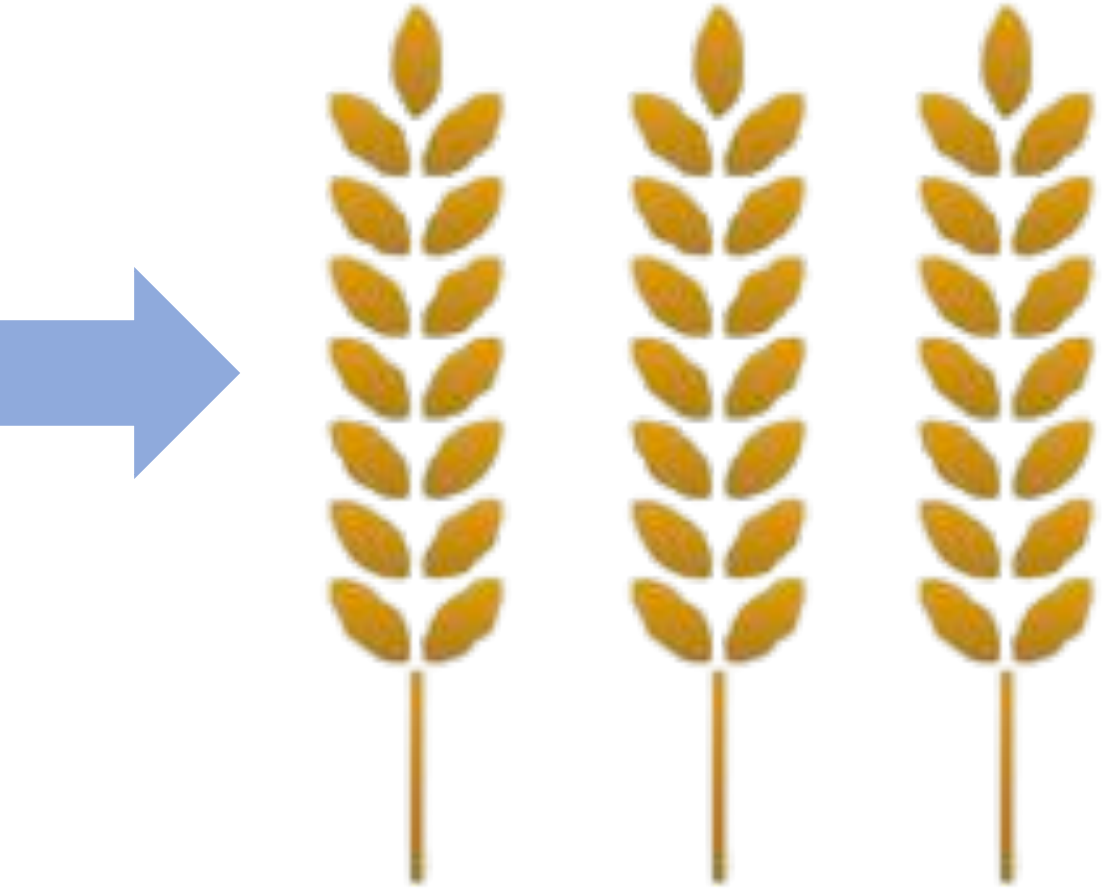
polia bajtov

Téma | topics



- len pridávame na koniec
- čítame zo začiatku

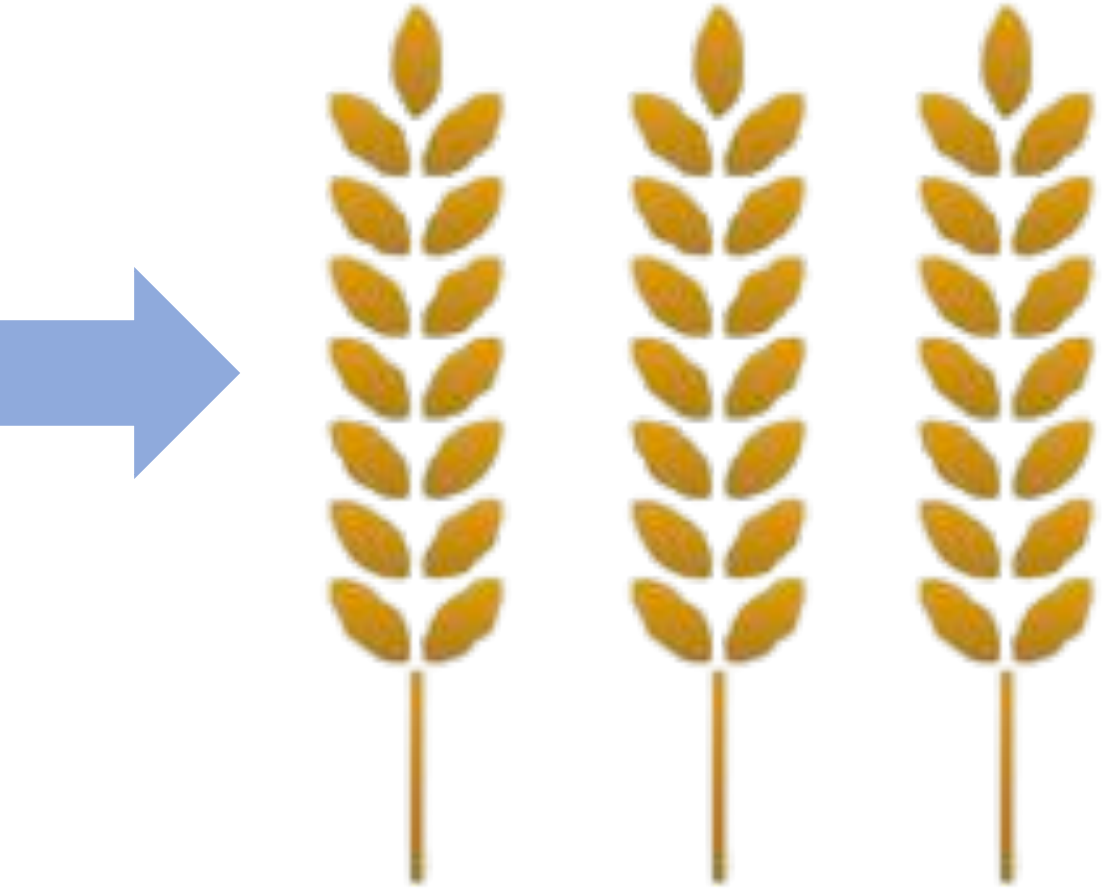
Konzumenti | Consumers



3 2 1

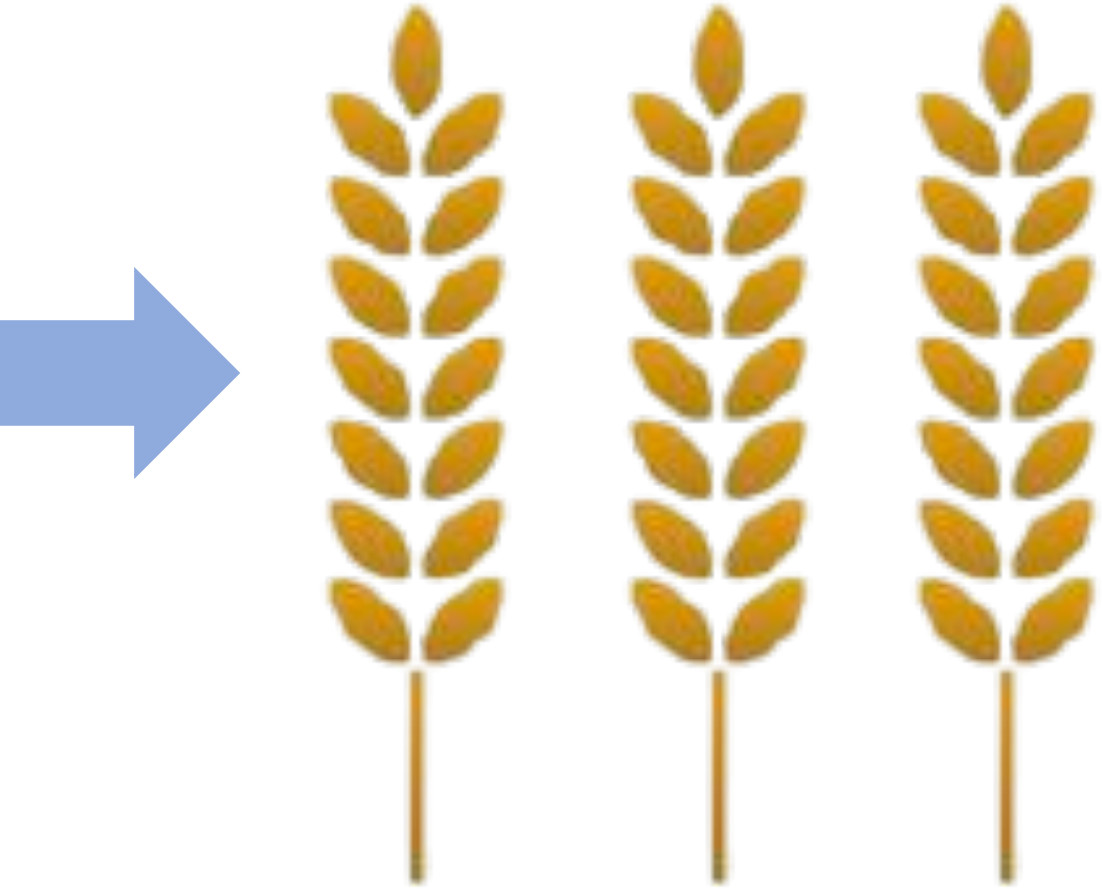
správy z topicu sa po konzumácii nestrácaju!

Konzumenti | Consumers



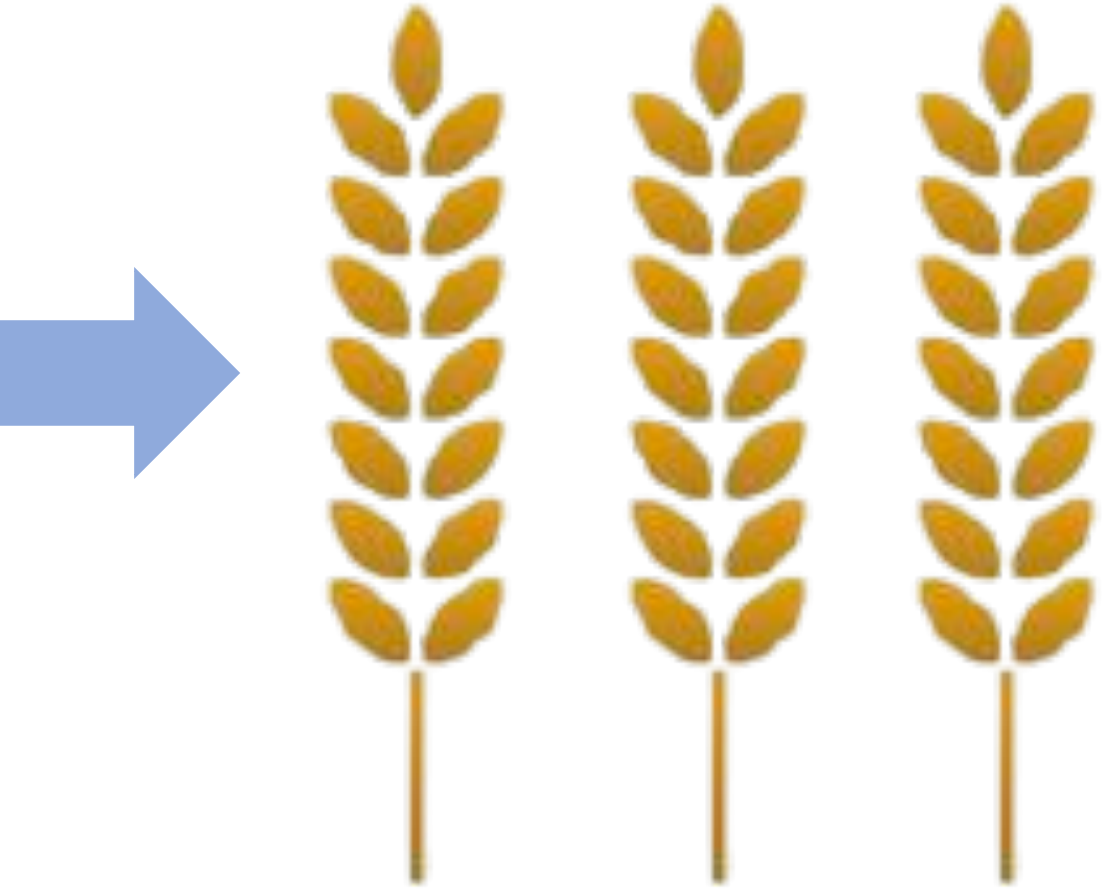
topic definuje pravidlá pre expiráciu správ

Konzumenti | Consumers



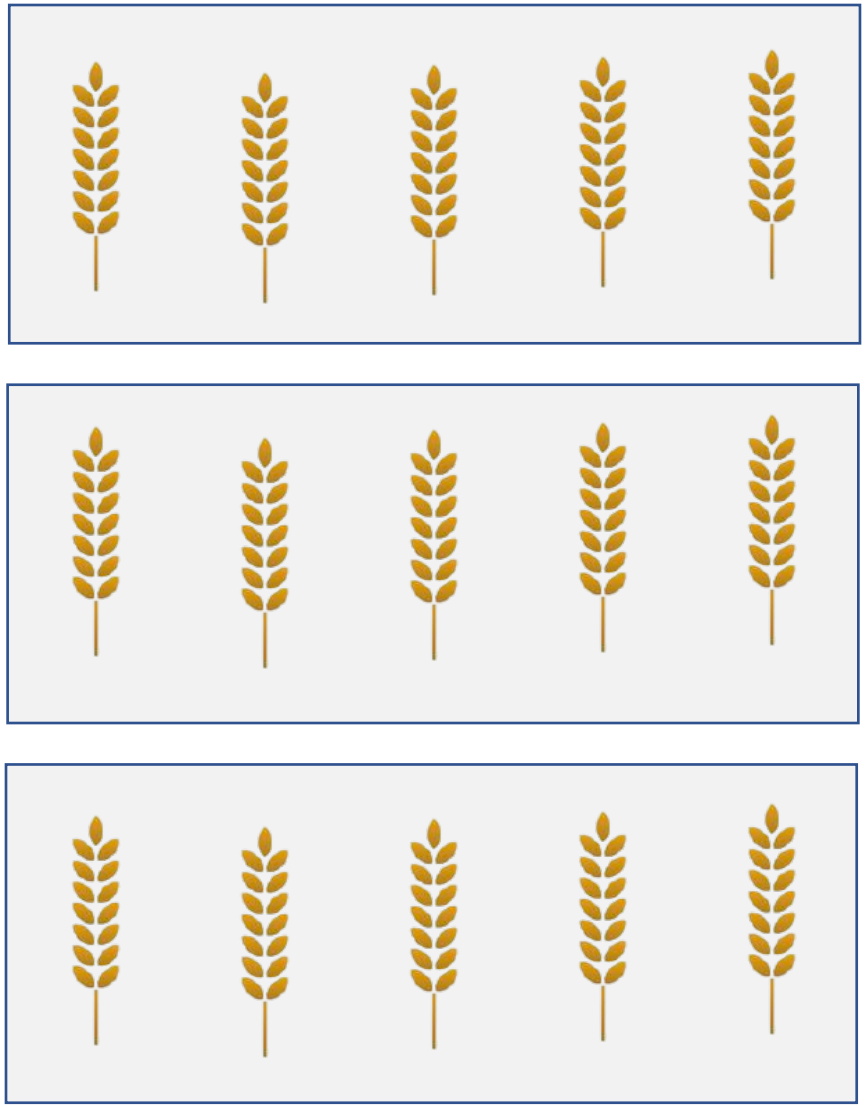
konzument môže čítať topic viacnásobne

Konzumenti | Consumers



konzument si pamätá offset, od ktorého začne čítať

Škálovanie topicov cez partície



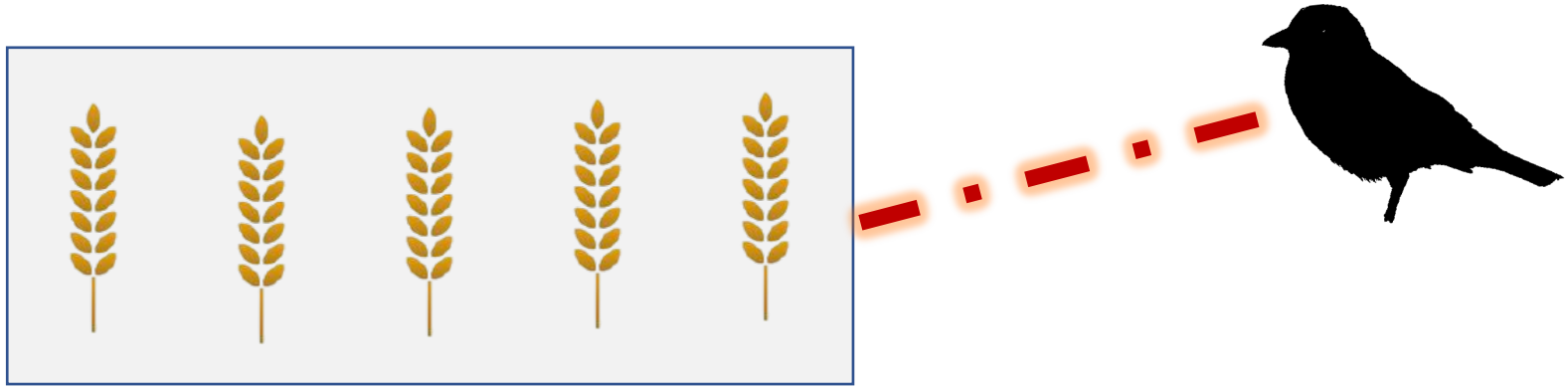
- topic rozporcujeme na časti
- časti rozhádžeme na uzly

Škálovanie konzumentov

Consumer Group: členovia kŕdľa súperia o správy

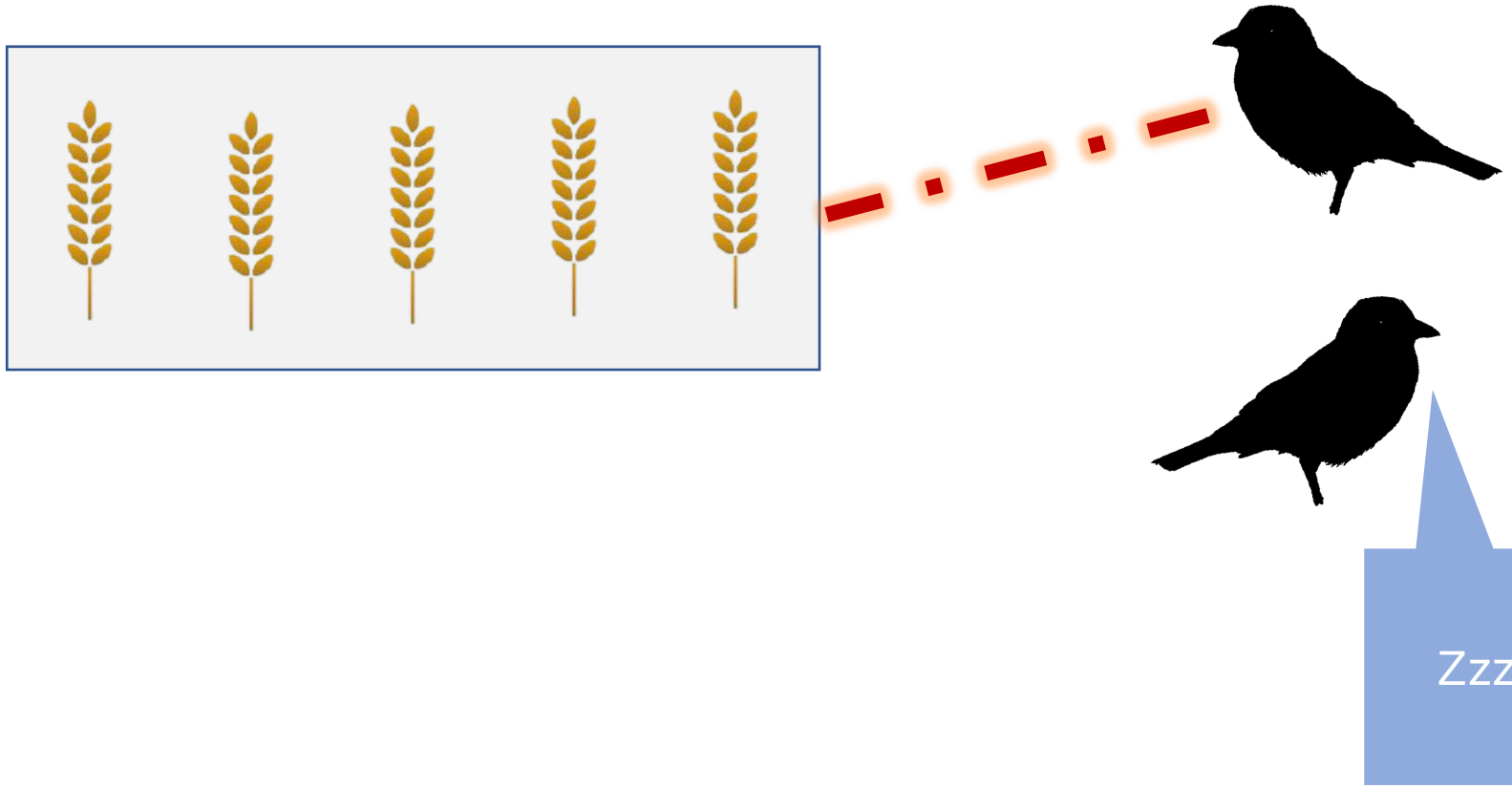
Príklad:

1 topic, 1 partícia,
1 consumer group, 1 člen



Príklad:

1 topic, 1 partícia,
1 consumer group, 2 členovia

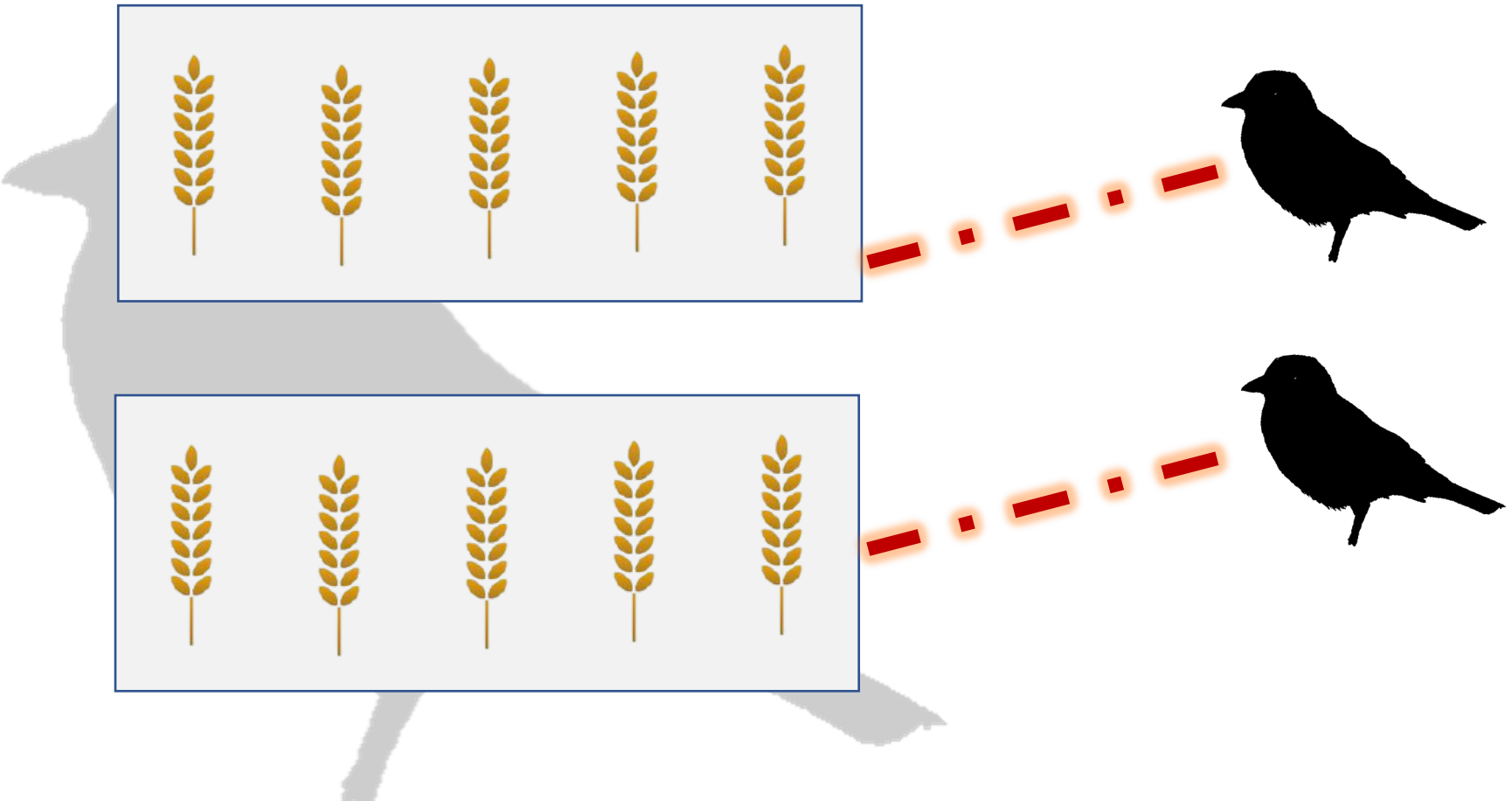


Pravidlo

Z 1 partície konzumuje najviac 1 člen consumer groupy

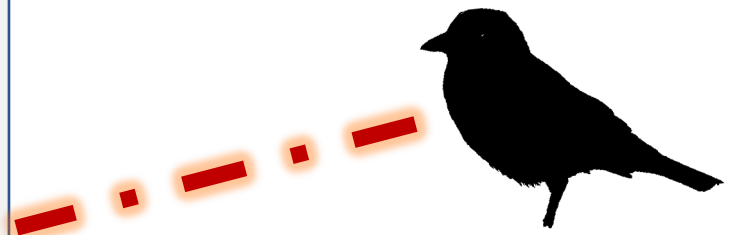
Príklad:

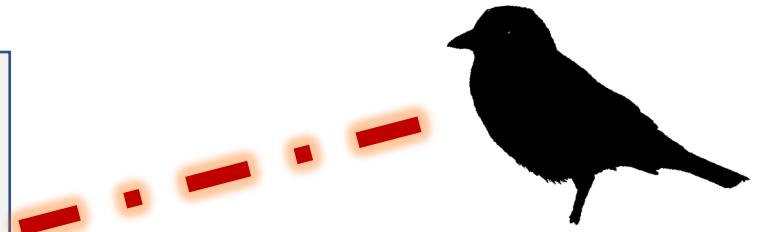
1 topic, 2 partície,
1 consumer group, 2 členovia



Príklad:

1 topic, 2 partície,
1 consumer group, 2 členovia





Delenie práce!
Súperenie!
Konkurencia!

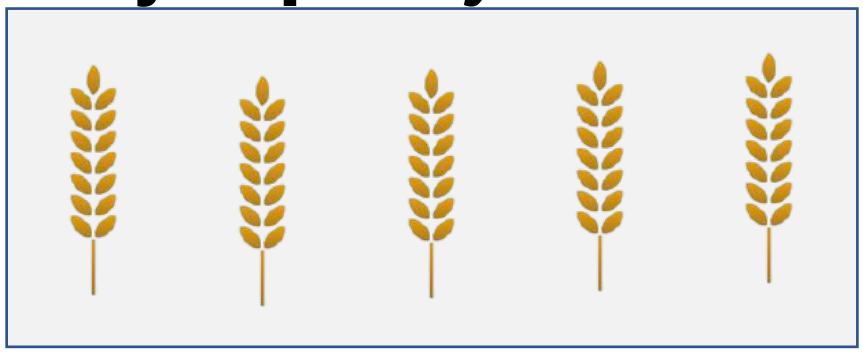
Producenti a partície

Kafka **rozumne** zadeľuje produkované správy do partícií

Kafka rozumne zadeľuje správy

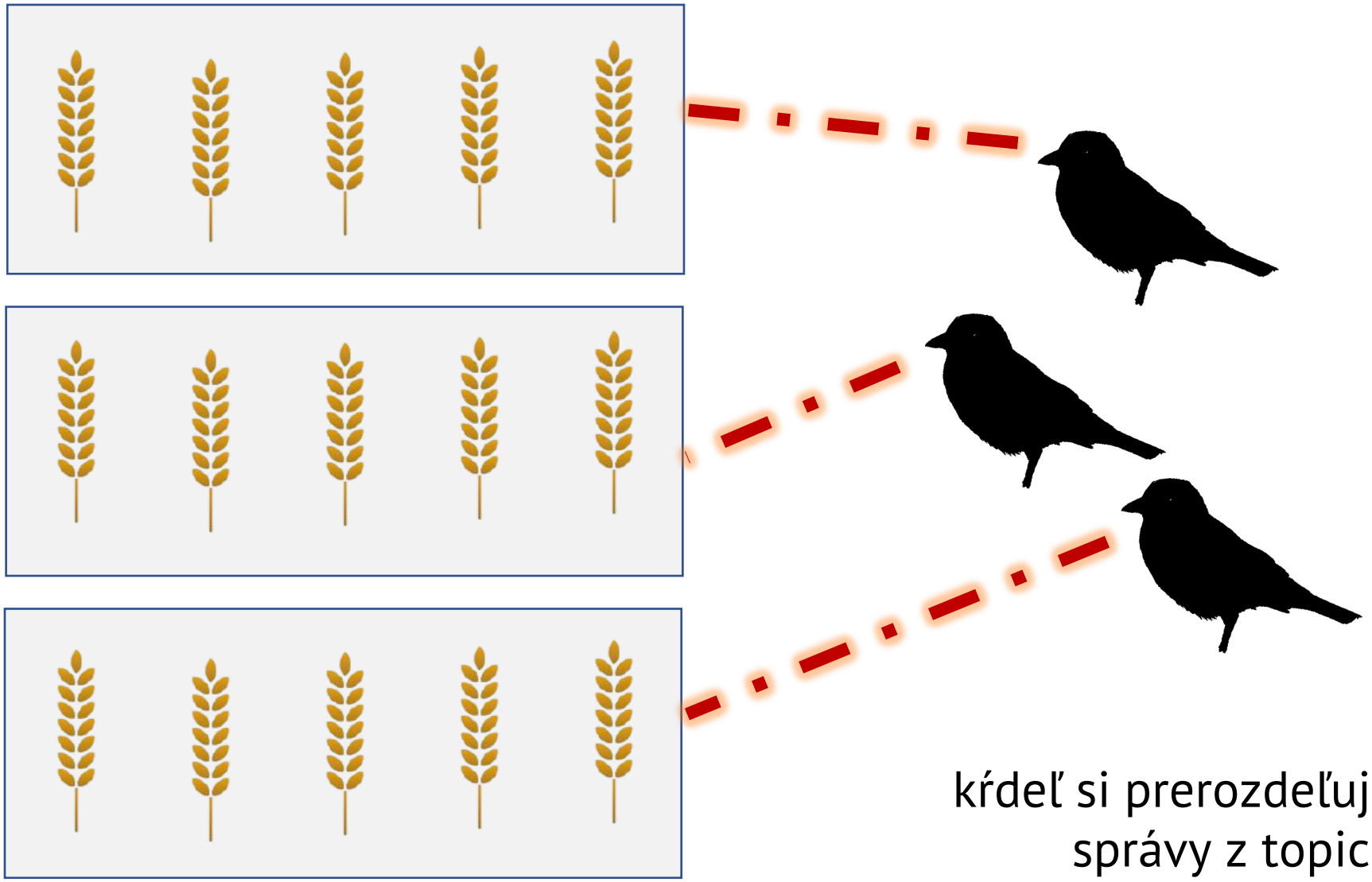
round robin: en-ten-týky

hash: vyráta sa odtlačok správy



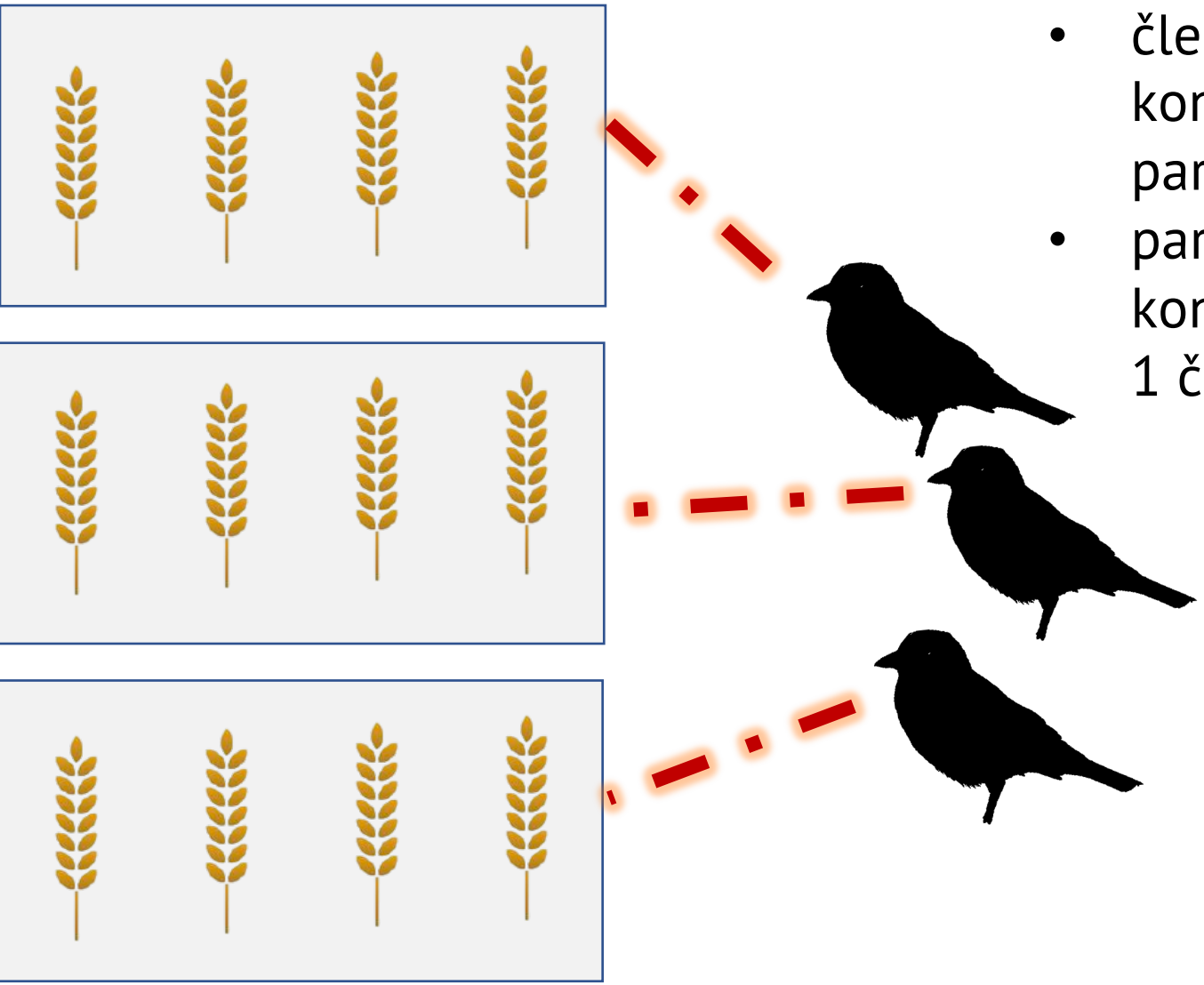
Príklad:

1 topic, 3 partície — 1 consumer group, 3 členovia



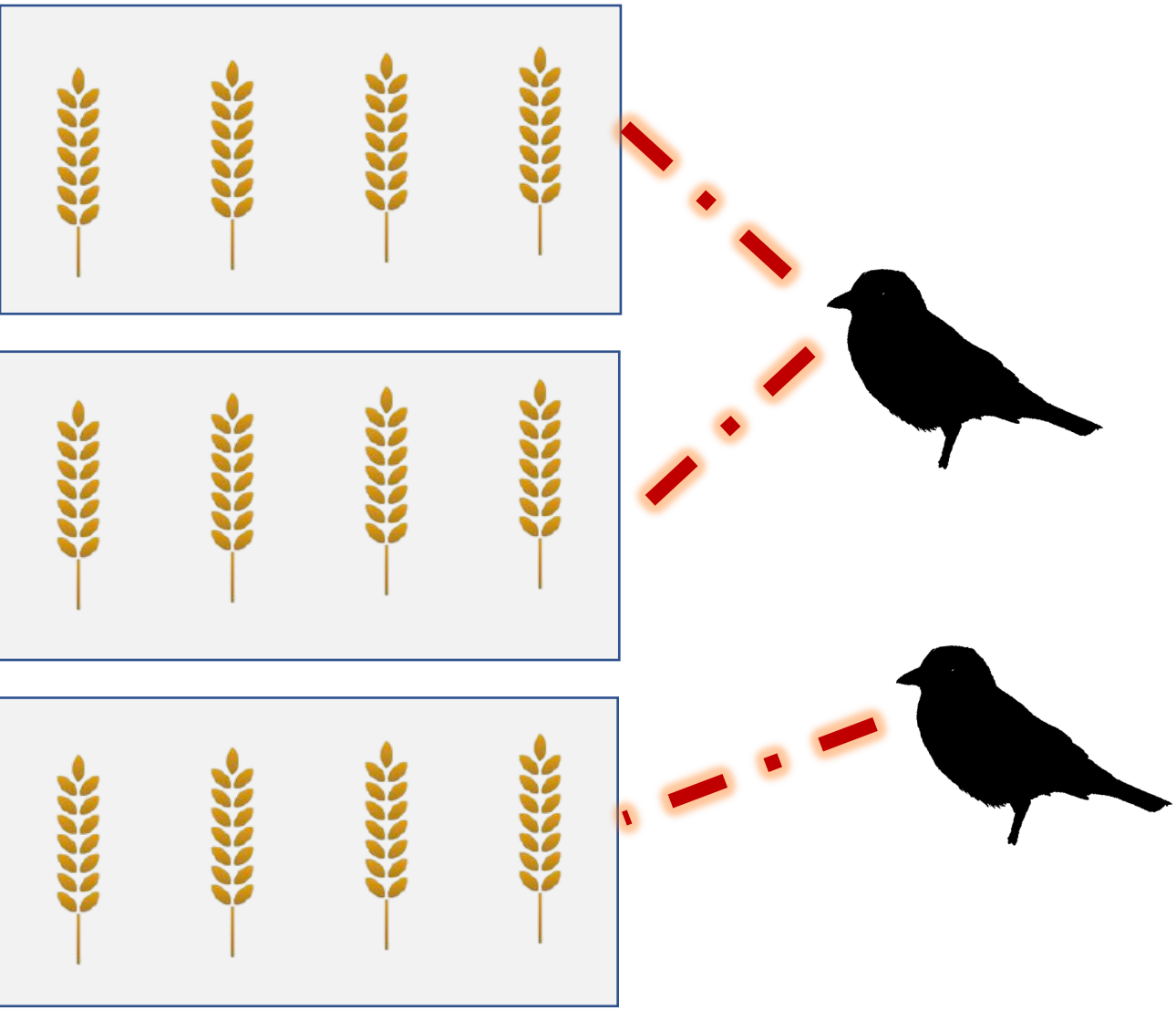
kŕdeľ si prerozdeľuje správy z topicu

Rebalancing

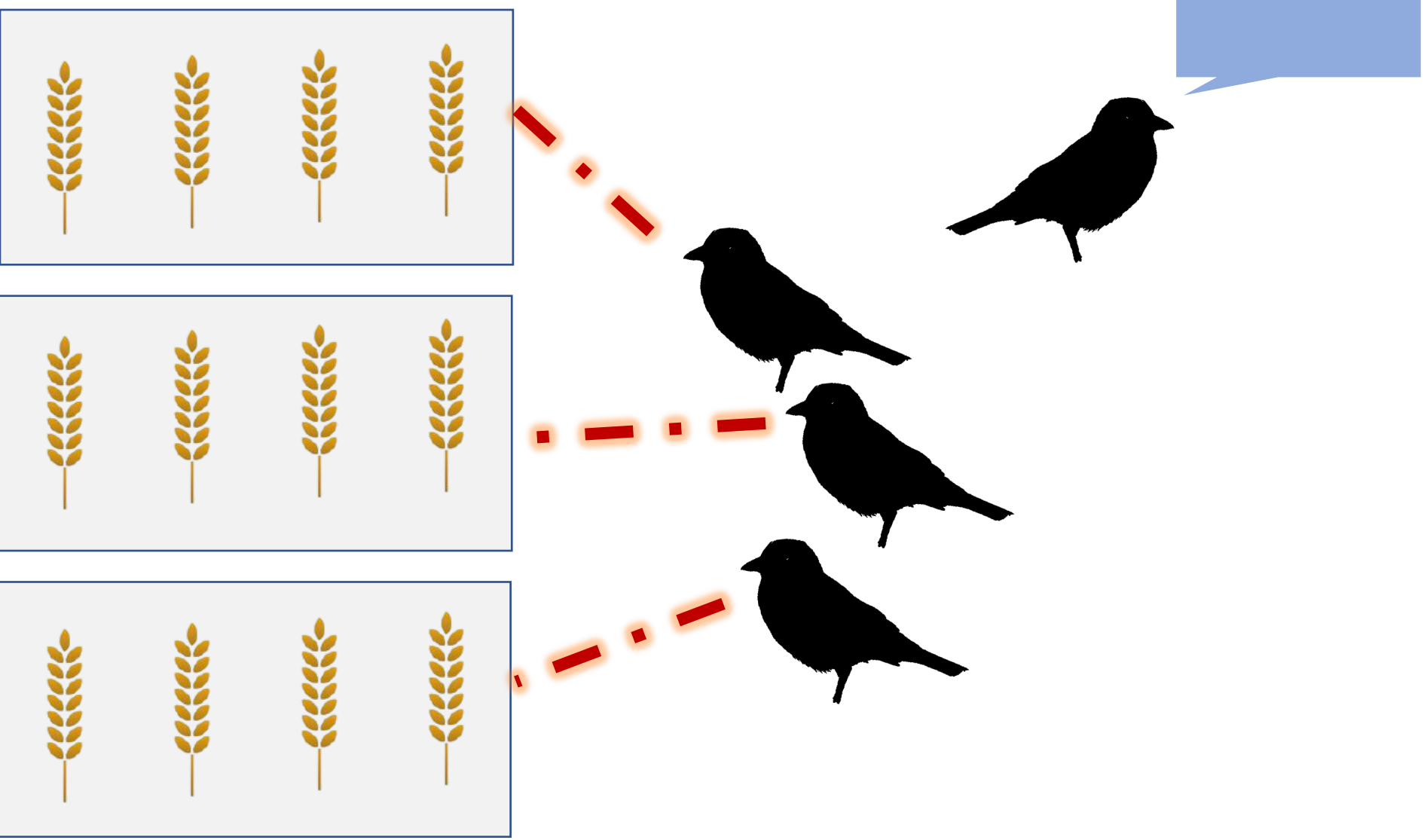


- člen skupiny konzumuje aspoň 1 partíciu
- partícia je konzumovaná najviac 1 členom

Partícií > konzumentov v skupine?



Partícií < konzumentov v skupine?



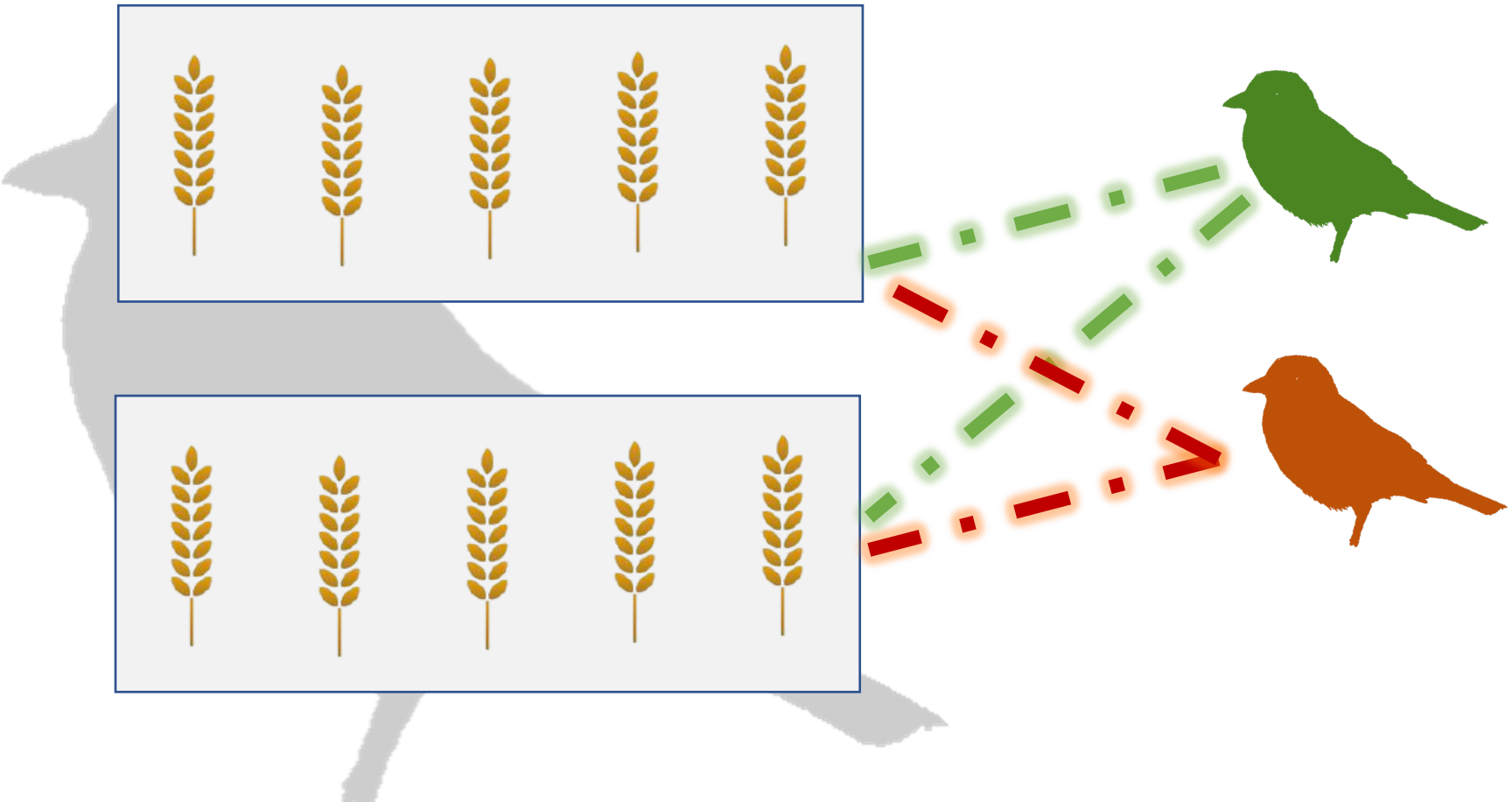
Partície = mechanizmus škálovania

- partície nastavujeme pri tvorbe topicu
- zväčšiť je možné, zmenšiť nie
- treba odhadnúť podľa tempa producentov a tempa konzumentov

Príklad:

1 topic, 1 partícia –

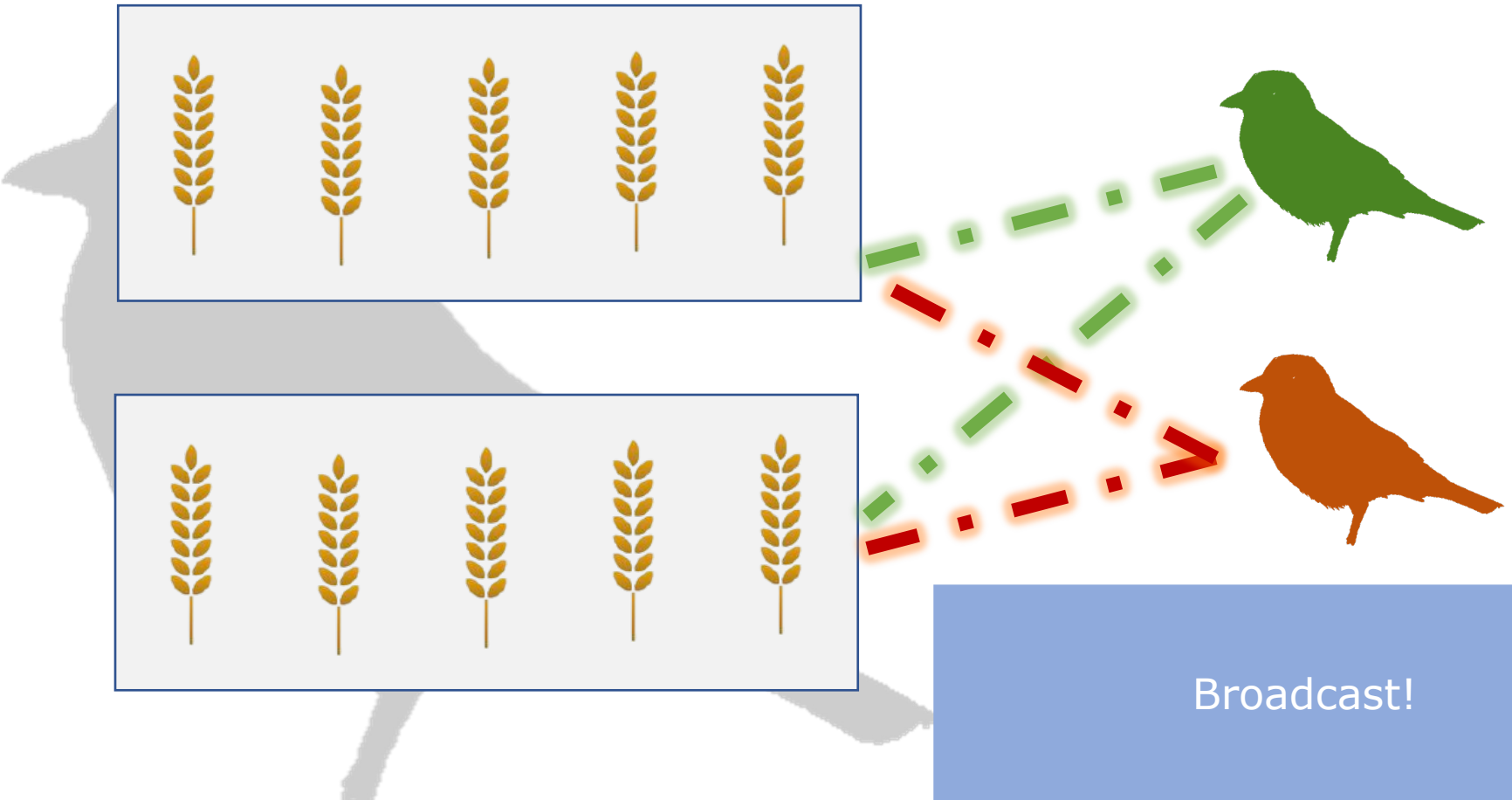
2 consumer groupy po 1 člene



Príklad:

1 topic, 1 partícia –

2 consumer groupy po 1 člene

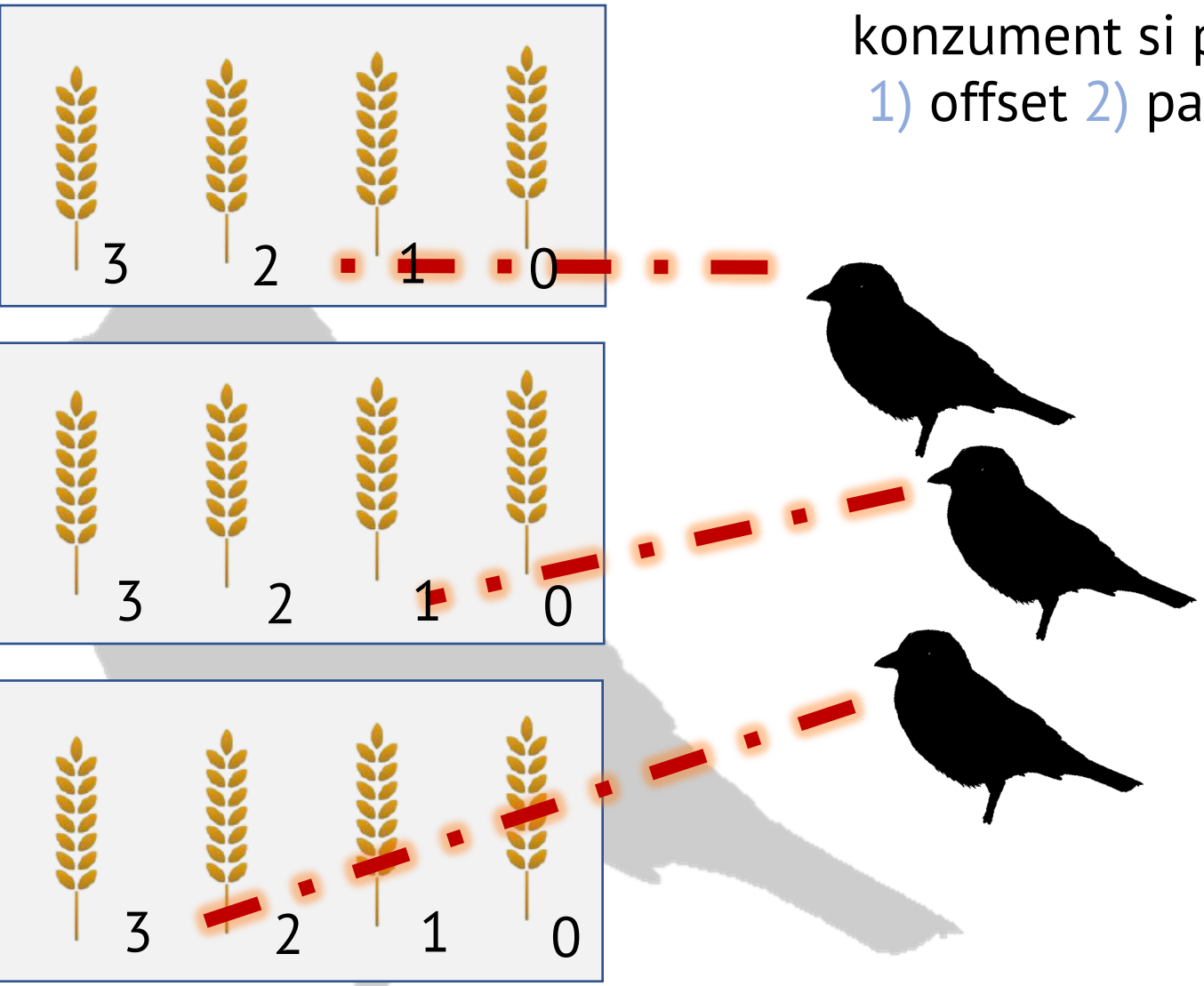


Broadcast!

Partície = mechanizmus škálovania

- správy doručené do jednej partície sú v poradí odoslania
- správy doručené do topicu môžu byť poprehadzované
  - môžu sa ocitnúť v rozličných partíciách
  - tým môžu byť konzumované v rozličnom poradí
- hint: nastaviť kľúč správy a particiovať podľa hashu

Konzum z partície



konzument si pamätá
1) offset 2) partície 3) topicu

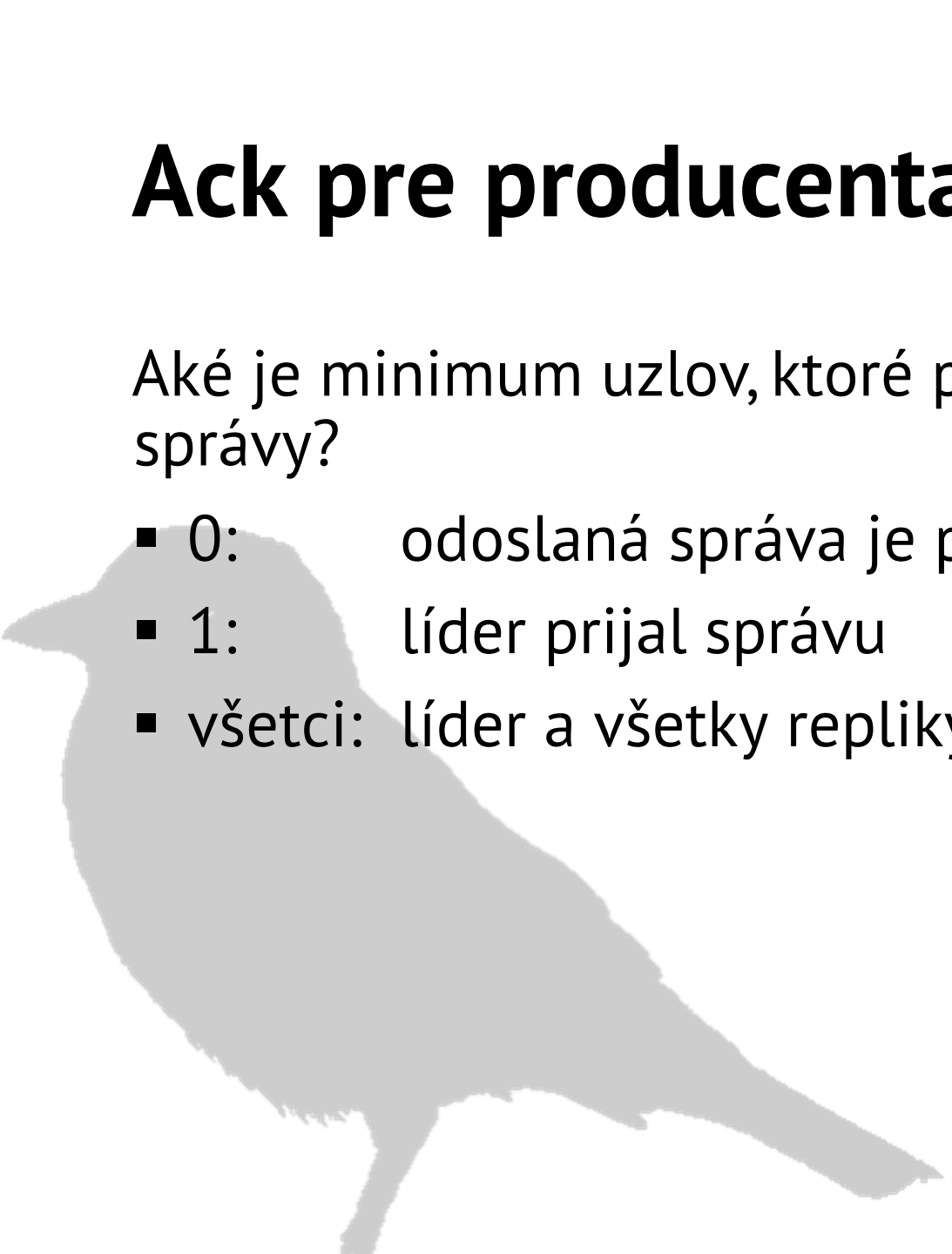
Výkonní a spoľahliví producenti

- Producenti môžu dostávať ACK od Kafky
- Tri úrovne kompromisu **výkon** vs **spoľahlivosť doručenia**

Ack pre producenta

Aké je minimum uzlov, ktoré potvrdí prijatie správy?

- 0: odoslaná správa je prijatá správa
- 1: líder prijal správu
- všetci: líder a všetky repliky prijali správu



Rýchli konzumenti

- Kafka nemá acknowledgement konzumentov
- „Tu máte partície topicu, čítajte si od ktorého indexu chcete“.

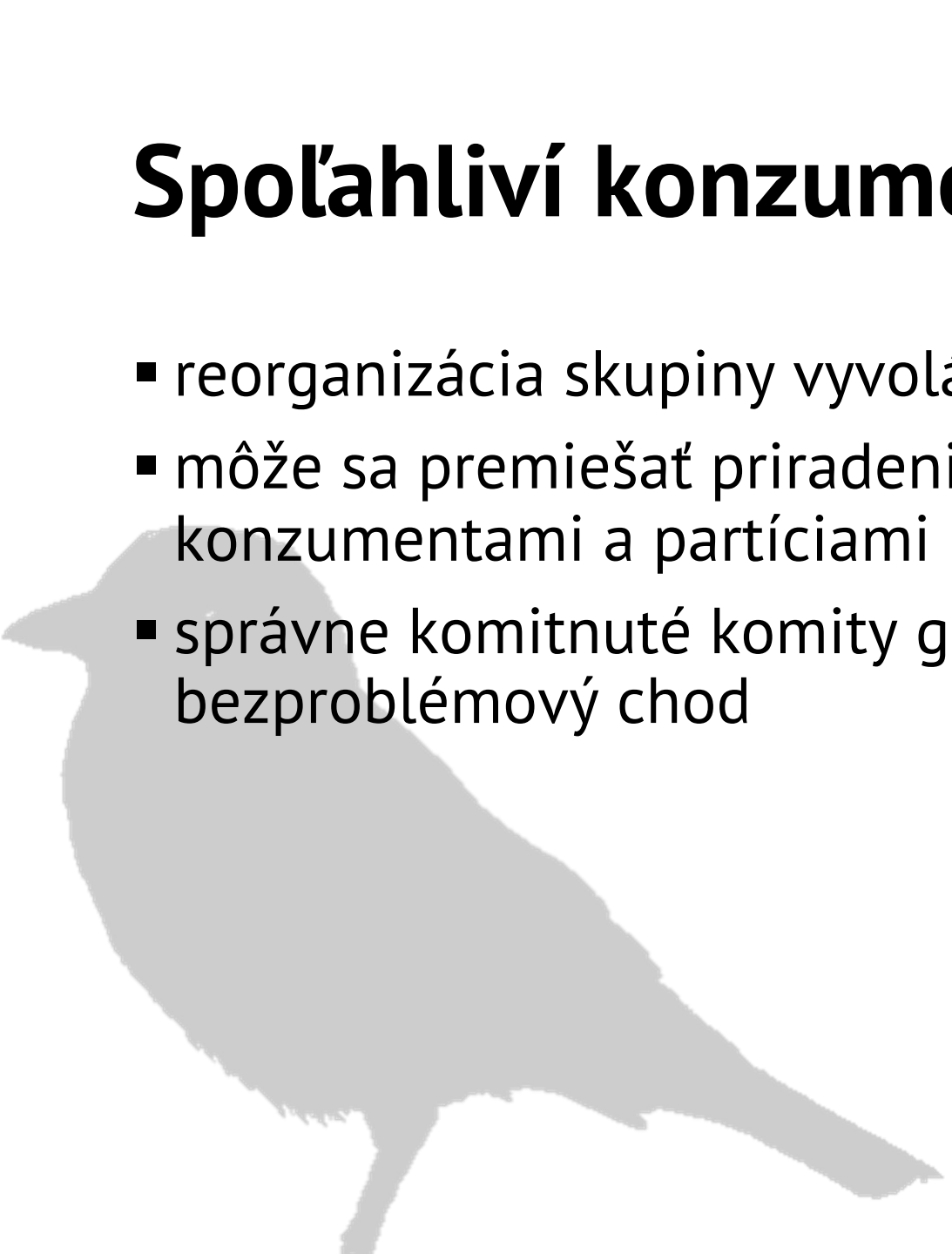
Spoľahliví konzumenti

- konzument v skupine môže **commitnúť** offset v partícii topicu

„Haló, Kafka?! Patrím do šedého kŕdľa a z tretej partície som spracoval správy po offset 3“

Spoľahliví konzumenti

- reorganizácia skupiny vyvolá **rebalans**!
- môže sa premiešať priradenie medzi konzumentami a partíciami
- správne komitnuté komity garantujú bezproblémový chod



At least once delivery

Konzument spracoval 5 správ, ale komitol len offset 2 (0, 1, 2)

”Komitol menej správ ako spracoval“

Po rebalanse: 2 správy sa spracujú viackrát

offsety 3 a 4

At most once delivery

Neporiadny konzument spracoval 2 správy, ale komitol offset 4 (0, 1, 2, 3, 4)

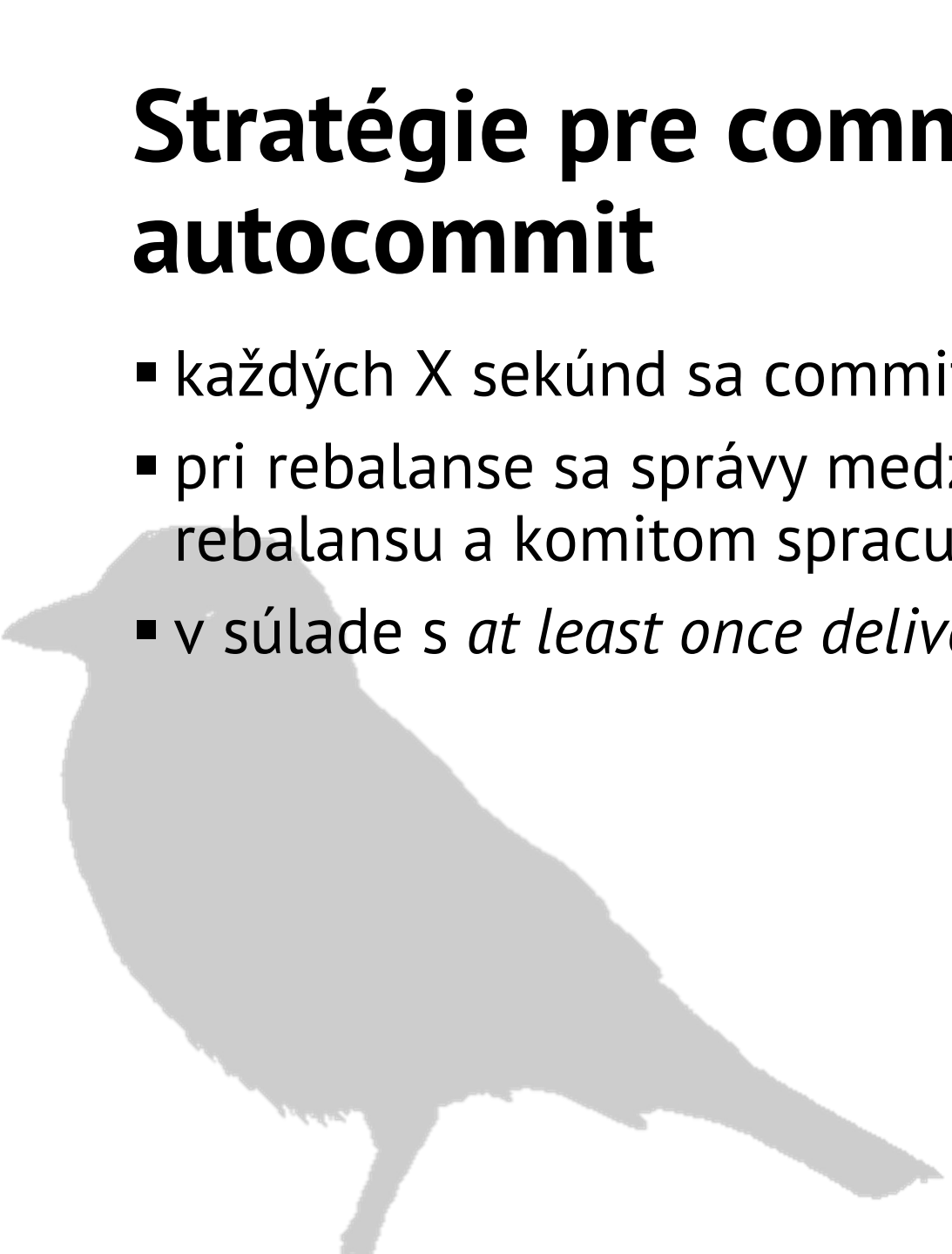
”Komitol viac správ než spracoval“

Po rebalanse: 3 správy sa „zmeškajú“

offsety 2, 3 a 4

Stratégie pre commit: autocommit

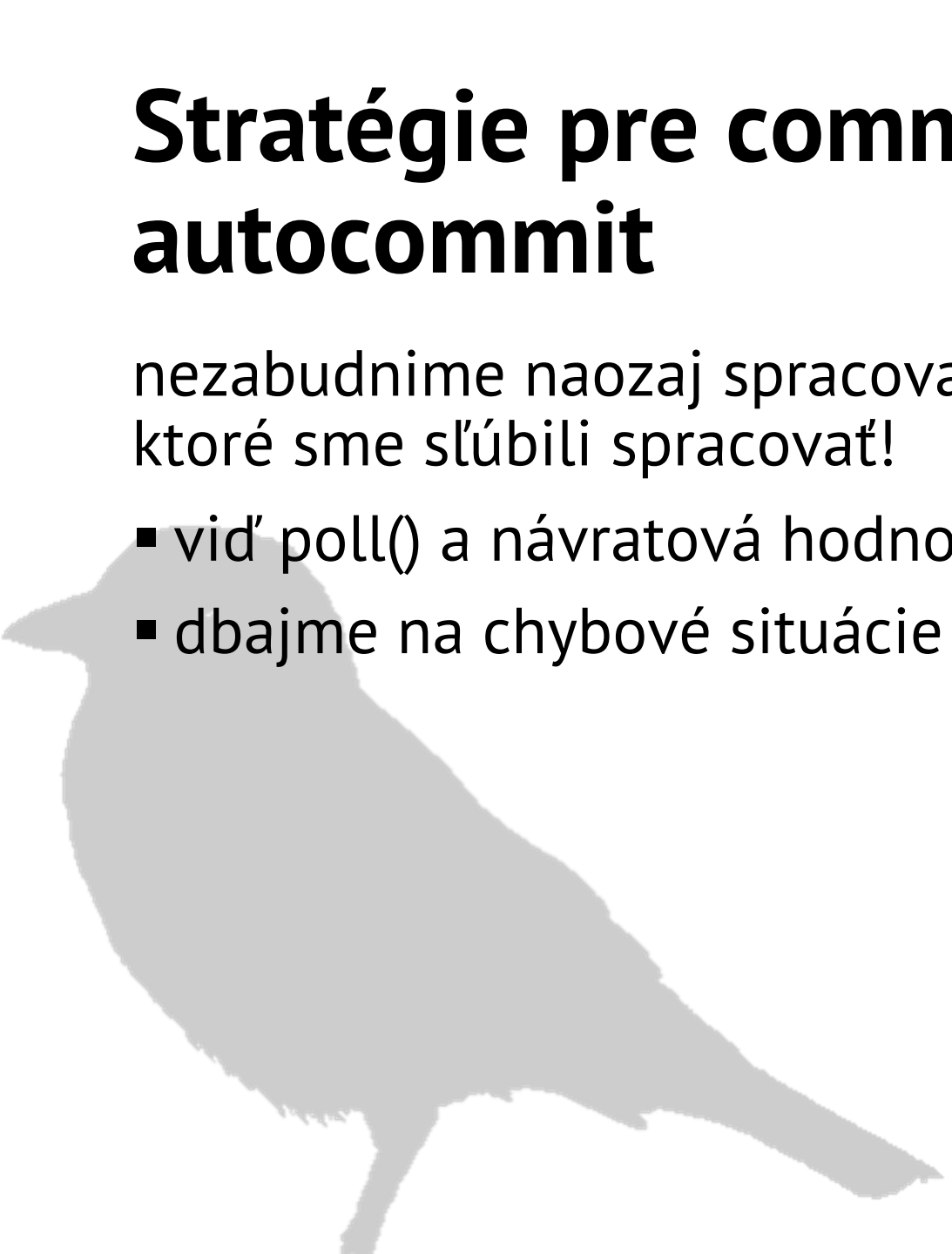
- každých X sekúnd sa commitne
- pri rebalanse sa správy medzi momentom rebalansu a komitom spracujú viackrát
- v súlade s *at least once delivery*



Stratégie pre commit: autocommit

nezabudnime naozaj spracovať všetky správy, ktoré sme sľúbili spracovať!

- viď poll() a návratová hodnota
- dbajme na chybové situácie pri spracovaní správ



Stratégie pre commit: ručný komit

- prečítame dávku správ
- spracujeme
- explicitne komitneme

- ak nastane počas dávky rebalans, celá dávka bude at least once

Stratégie knižníc

- knižnice ponúkajú rozumné nadstavby
- Spring Boot + Kafka: inteligentné komitovanie podľa viacerých kritérií

Ďalšie benefity Kafky

- exactly once-delivery
  - preskúmané a funkčné
- podpora pre schému správ
  - JSON / Avro / Protobuf

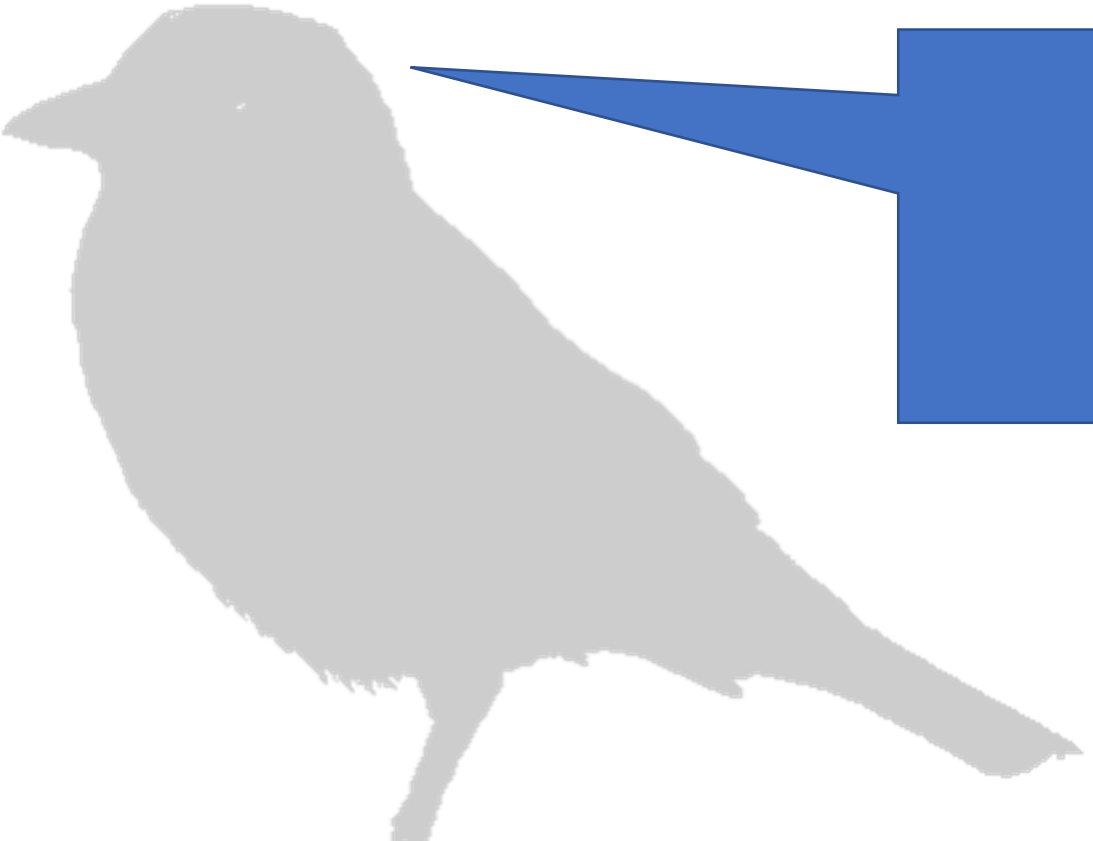
Kafka vs RabbitMQ

Kafka: "dumb broker, smart consumers"

- menej out of box možností, viac výkonu
- výkonnosť je ovplyvnená architektúrou a implementáciou projektu
- implementácia nad JVM

RabbitMQ: "smart pipes, dumb consumers"

- typické topológie sú out of box
- stačí pochopiť architektúru a správne ju použiť
- implementácia je potom jednoduchšia



Otázky?