Failure Detection and Membership II

ECE 599 / CS 519 – SPRING 2015

Failure Detector Properties …

- Completeness
- Accuracy
- Speed
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

…Are application-defined Requirements

- Time to first detection of a failure

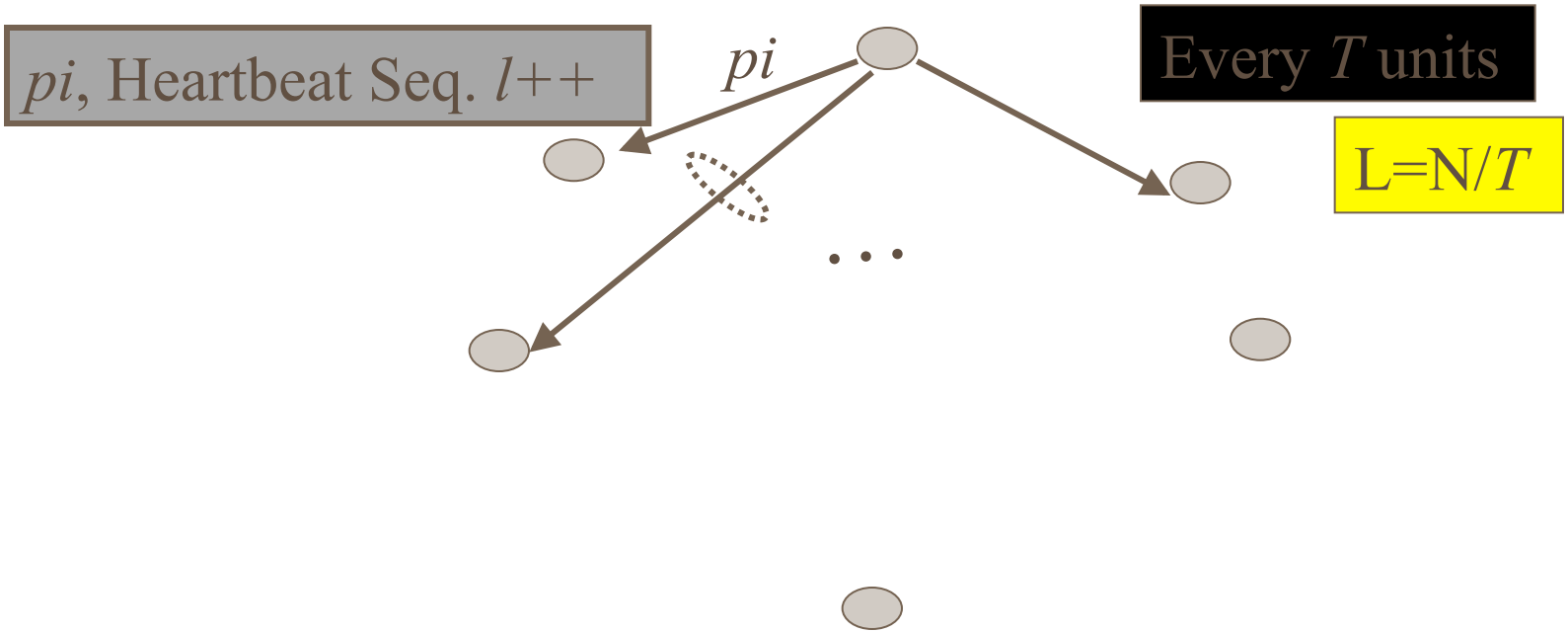
- Scale
  - Equal Load on each member
  - Network Message Load

…Are application-defined Requirements

- Completeness → Guarantee always
- Accuracy → Probability *PM(T)*
- Speed → *T* time units
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

L = messages per time unit: Compare this across protocols

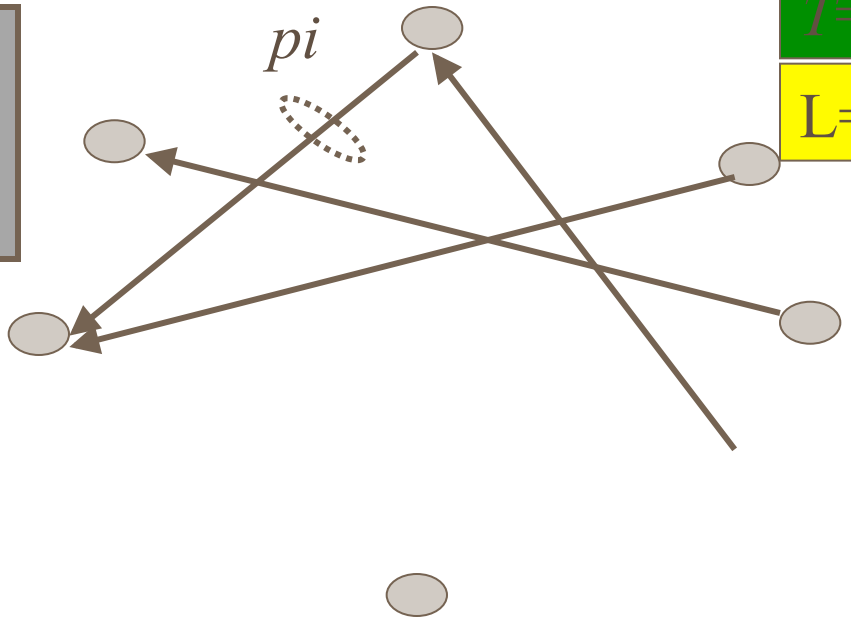
All-to-All Heartbeating



Gossip-style Heartbeating

Array of Heartbeat Seq. *l* for member subset

Every tg units =gossip period, send O(N) gossip message



T=logN * tg

L=1/tg=logN/*T*

What's the Best/Optimal we can do?

- *Worst case* load L* per member in the group (messages per time unit)
  - as a function of *T*, *PM(T)*, N
  - Independent Message Loss probability p_{ml}

$$L^* = \frac{\log(PM(T))}{\log(p_{ml})} \cdot \frac{1}{T}$$

Heartbeating

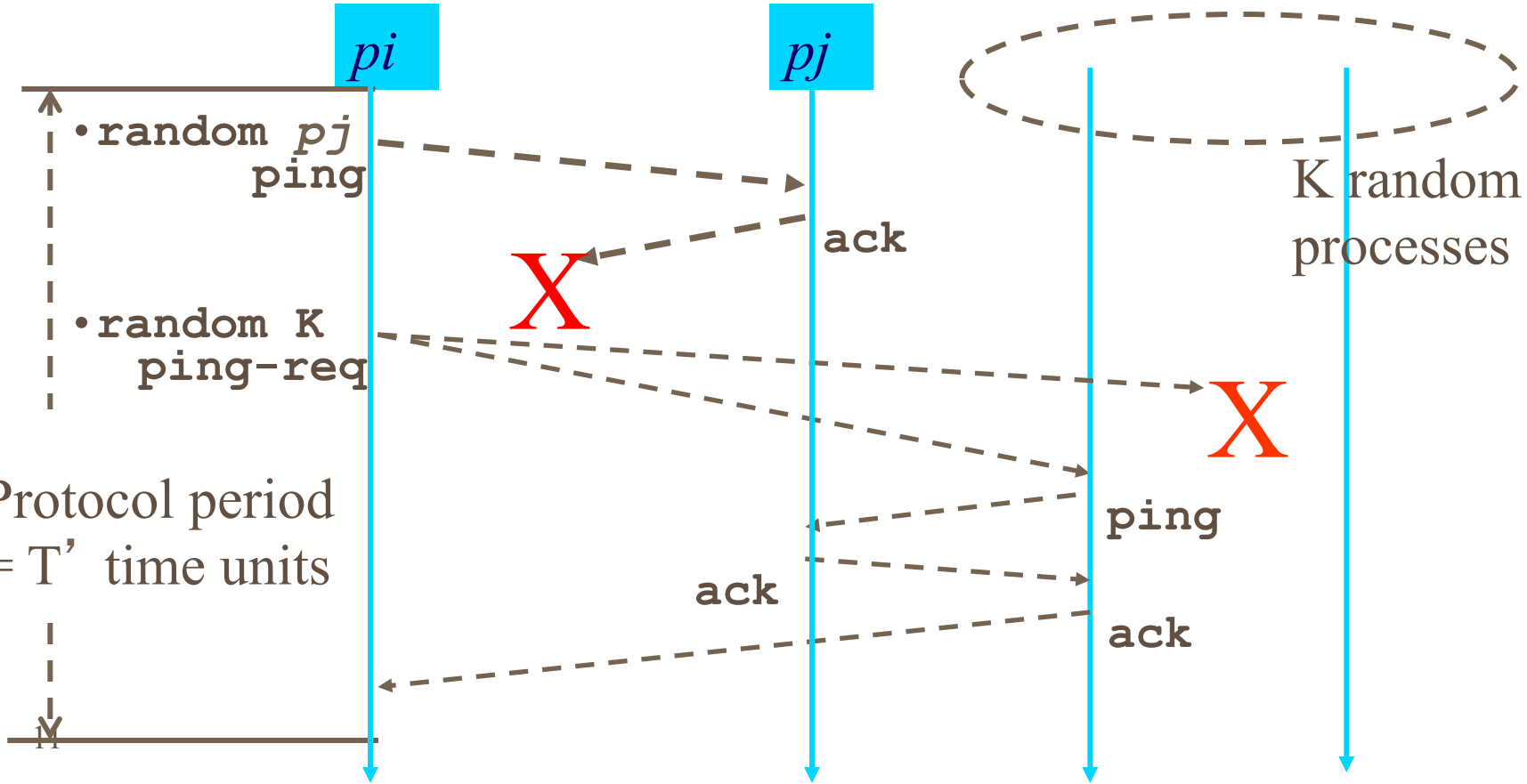
- Optimal L is independent of N (!)
- All-to-all and gossip-based: sub-optimal
  - L=O(N/T)
  - try to achieve simultaneous detection at ***all*** processes
  - fail to distinguish *Failure Detection* and *Dissemination* components

- Key:
  - Separate the two components
  - Use a non heartbeat-based Failure Detection Component

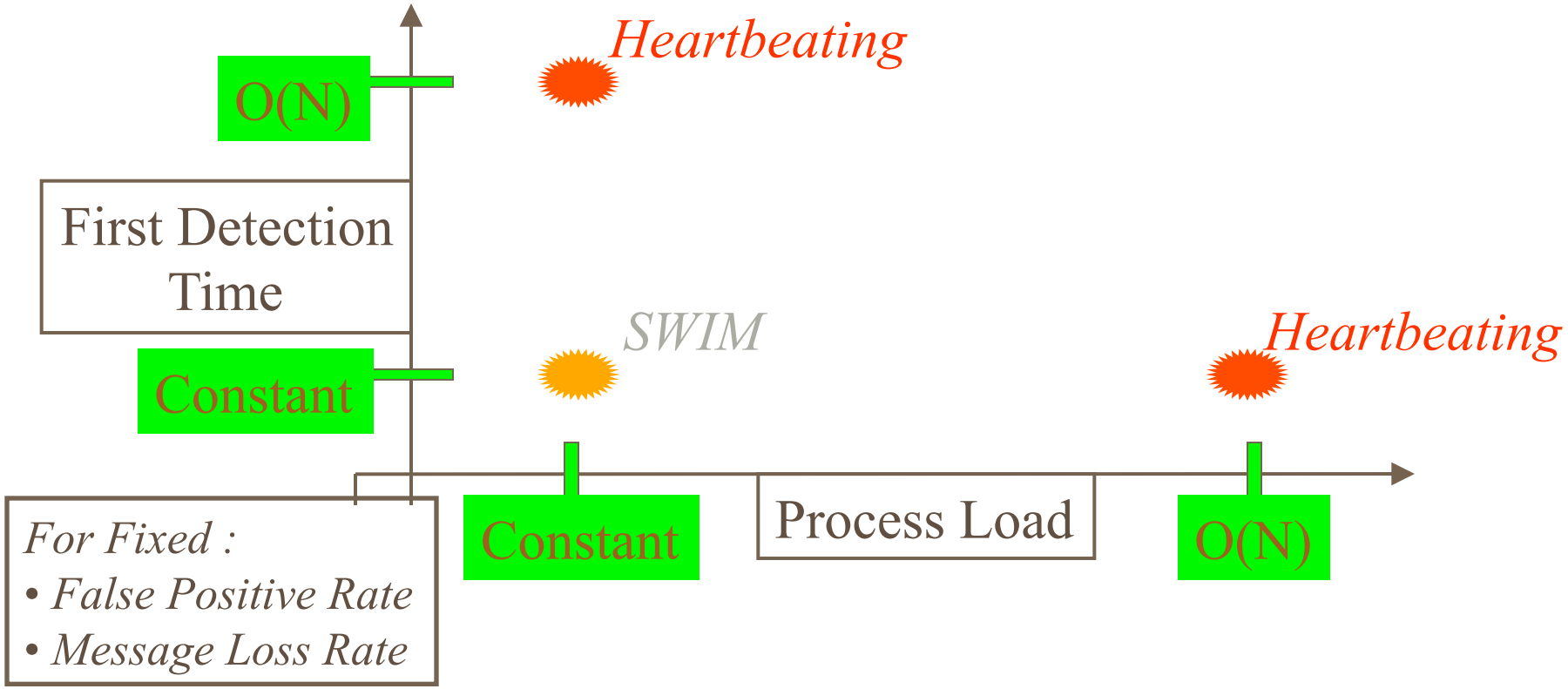
Next

- Is there a better failure detector?

SWIM Failure Detector Protocol



SWIM versus Heartbeating



SWIM Failure Detector

Parameter	SWIM
First Detection Time	• Expected $\left[\frac{e}{e-1}\right]$ periods • Constant (independent of group size)
Process Load	• Constant per period • < 8 L* for 15% loss
False Positive Rate	• Tunable (via K) • Falls exponentially as load is scaled
Completeness	• Deterministic time-bounded • Within O(log(N)) periods w.h.p.

Accuracy, Load

PM(T) is exponential in -*K*. Also depends on *pml* (and *pf*)

- See paper

$$\frac{L}{L^*} < 28$$

$$\frac{E[L]}{L^*} < 8$$

for up to 15 % loss rates

Detection Time

Prob. of being pinged in T'= $1-(1-\frac{1}{N})^{N-1} = 1-e^{-1}$

E[T] = $\mathrm{T}'.\frac{e}{e-1}$

Completeness: *Any* alive member detects failure

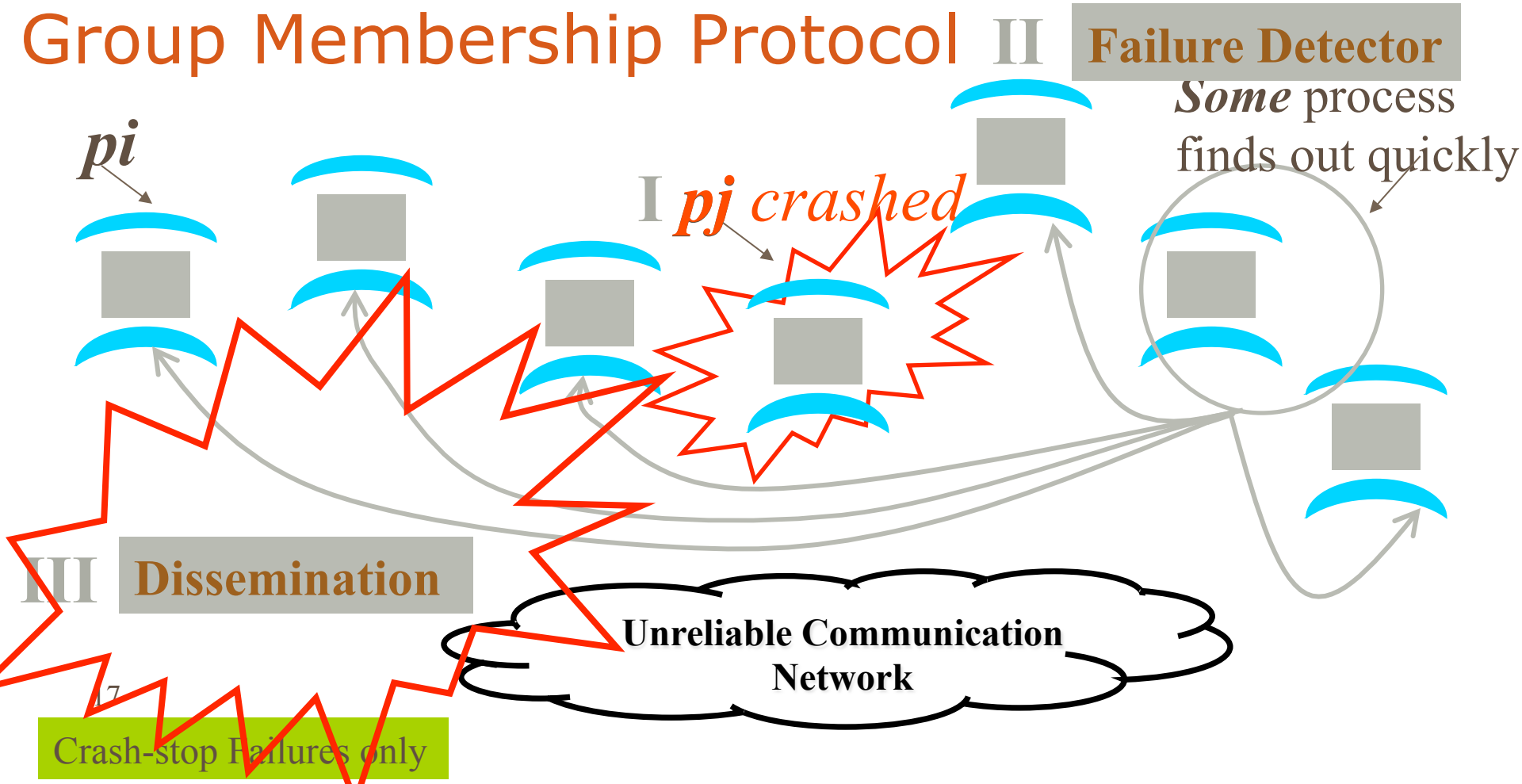
- Eventually
- By using a trick: within worst case *O(N)* protocol periods

Next

- How do failure detectors fit into the big picture of a group membership protocol?

- What are the missing blocks?

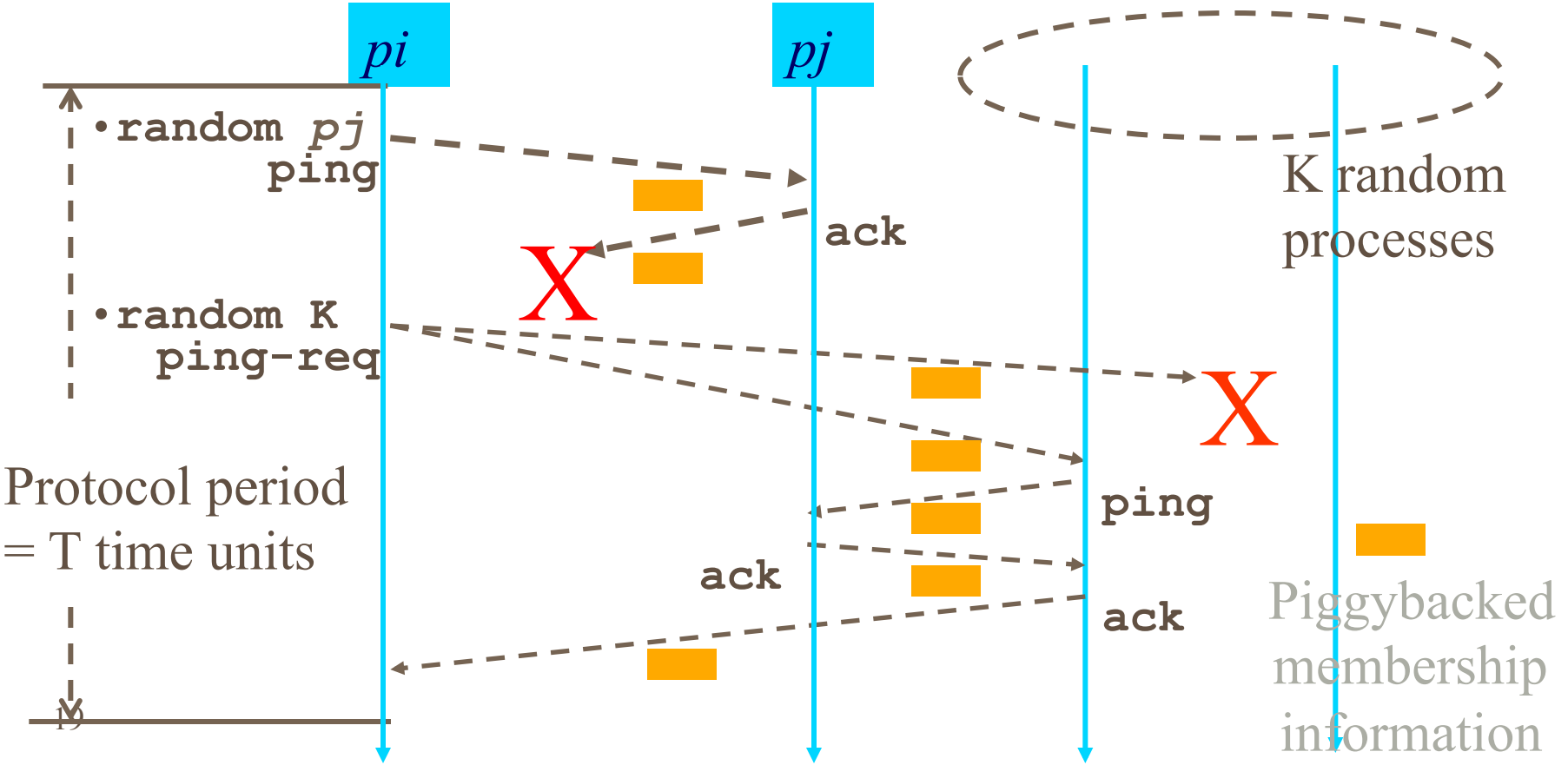
Group Membership Protocol



Dissemination Options

- Multicast (Hardware / IP)
  - unreliable
  - multiple simultaneous multicasts
- Point-to-point (TCP / UDP)
  - expensive
- Zero extra messages: Piggyback on Failure Detector messages
  - Infection-style Dissemination

Infection-style Dissemination



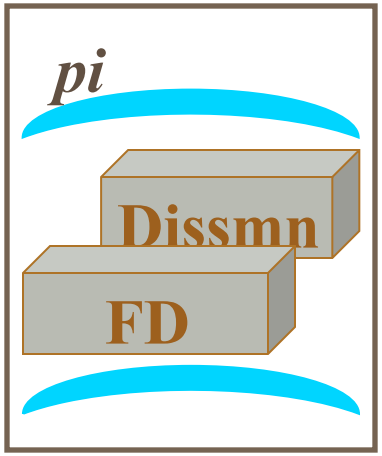
Infection-style Dissemination

- Epidemic/Gossip style dissemination
  - After $\lambda.\log(N)$ protocol periods, $N^{-(2\lambda-2)}$ processes would not have heard about an update
- Maintain a buffer of recently joined/evicted processes
  - Piggyback from this buffer
  - Prefer recent updates
- Buffer elements are garbage collected after a while
  - After $\lambda.\log(N)$ protocol periods, i.e., once they've propagated through the system; this defines weak consistency

Suspicion Mechanism

- False detections, due to
  - Perturbed processes
  - Packet losses, e.g., from congestion
- Indirect pinging may not solve the problem
- Key: *suspect* a process before *declaring* it as failed in the group

Suspicion Mechanism



Dissmn (Suspect *pj*)

Suspected

FD:: *pi* ping failed
Dissmn::(Suspect *pj*)

FD::*pi* ping success
Dissmn::(Alive *pj*)

Time out

Alive

Failed

Dissmn (Alive *pj*)

Dissmn (Failed *pj*)

Suspicion Mechanism

- Distinguish multiple suspicions of a process
  - Per-process *incarnation number*
  - *Inc #* for *pi* can be incremented only by *pi*
    - e.g., when it receives a (Suspect, *pi*) message
  - Somewhat similar to DSDV
- Higher inc# notifications over-ride lower inc#'s
- Within an inc#: (Suspect inc #) > (Alive, inc #)
- (Failed, inc #) overrides everything else

Wrap Up

- Failures the norm, not the exception in datacenters
- Every distributed system uses a failure detector
- Many distributed systems use a membership service
- Ring failure detection underlies
  - IBM SP2 and many other similar clusters/machines
- Gossip-style failure detection underlies
  - Amazon EC2/S3 (rumored!)