Welcome to Ubiquitous Computing, CSI 660

Prof. William A. Maniatty
Lecture 1

maniatty@cs.albany.edu

`http://www.cs.albany.edu/~maniatty/teaching/ubicomp/`

Welcome to Ubiquitous Computing (aka Pervasive Computing).

- So what is Ubiquitous Computing?
  - ▷ Immerses computers in a real environment
  - ▷ Sensors support interact with and control the environment.
  - ▷ Limited power supply, storage, memory and bandwidth.
  - ▷ Operate unattended (much like embedded systems).
  - ▷ Devices are mobile/wireless.
  - ▷ May reside on a person (wearable computing).
  - ▷ Have special peripherals.
  - ▷ Contrast this with virtual reality which immerses humans in a computer generated artificial environment.
- What are the Goals of This Course?
  - ▷ Prepare researchers
  - ▷ Learn about this area together
  - ▷ Try to find an opportunity to learn by doing
- Grading - See Syllabus
  - ▷ Projects (1) - 40
  - ▷ Exams (2) - 30 %
  - ▷ Reports (2 oral, 1 written) - 30 %

Administrative Stuff - Course Materials

Text Books - These are useful for background material

- Security for Ubiquitous Computing by Frank Stajano. John Wiley and Sons, Ltd. Wiley Series in Communications Networking & Distributed Systems, 2002.
  ISBN: 0-470-84493-0.
- Wireless Communications and Networks by William Stalling. Prentice Hall, 2002.
  ISBN: 0-13-040864-6.

Course Home Page:

`http://www.cs.albany.edu/~maniatty/teaching/ubicomp/`

Administrative Stuff - Course Policies

We are looking for research topics.

Goal: Reward good students

- So be good!
- Otherwise it is possible to fare poorly.

Class covers key concepts, you'll need to read on your own.

Learn by doing and reading, don't just sit and listen.

Please attend.

- Otherwise you'll miss out
- Your grades may reflect that.

Number of talks and scope of projects depend on enrollment.

Grading gripes - I regrade the entire item, not just the complaint

- On Exam - hand back exam before leaving class with a note about grading issues
- On Projects / Homeworks - Must be within one week of the return.

Historical Origins and Trends

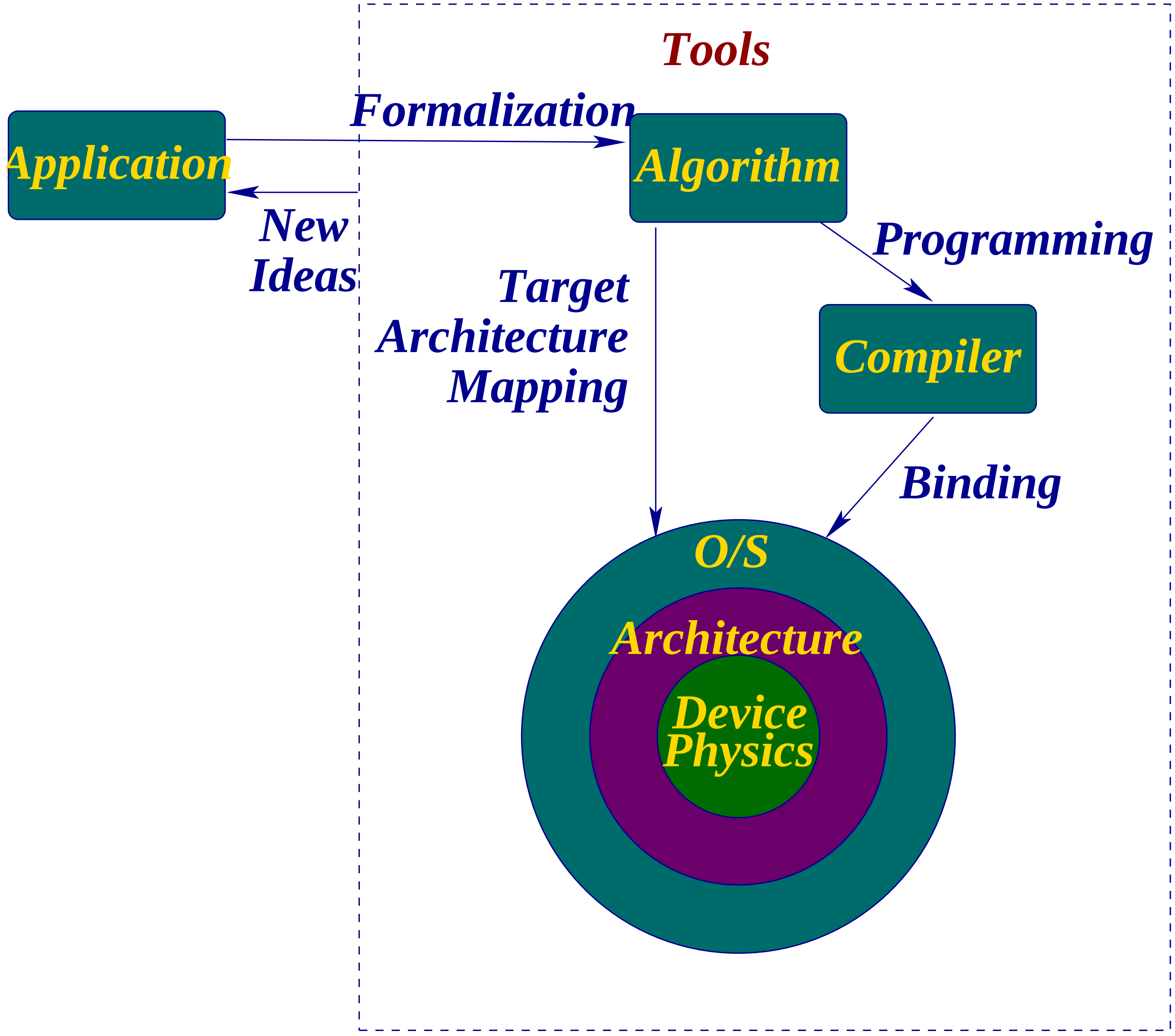
Computers are becoming smaller and cheaper over time

- Originally few computers many operators
  - ▷ Machines Expensive and Large
  - ▷ People (relatively) cheap
- Trend toward more computers per person
  - ▷ Users may not be tech savvy
  - ▷ Even tech savvy users have limited time
  - ▷ Minimal intervention is required

People don't want to be separated from their data

- But spying on users upsets them
- And can violate laws - security is important
- Mobility and wireless access are critical.

Tool/Application Interactions



Architecture Features and Trends

Component	Properties	Trends/Issues
Processor	Electronic	Moore's Law, Power/Speed Trade off
Memory	Electronic	Moore's Law, Power/Speed Trade off
Persistent Storage	Electromechanical	MEMS/NVRAM
Networking	Electronic	Signal Strength, Encoding Security
User Interface Peripherals	Electromechanical	sensory limitations
Power Supply	Chemical	Very limiting! Fuel Cells?
Software	Embedded	Stand Alone, Resource limited

Observations/My Opinions

Architectural trends seem more clear

Some user motivations/trends observed

- Young kids in the mall deploy new cheap technology
- Mobile devices and Cell Phones beginning to merge

New small machines feel like old version of previous generation

- Small Memory
- Limited Processing
- Limited Connectivity
- Big Difference - Limited Power

However, still looking for killer apps.

- Requires identifying a need
- Reflects what people want to do.

Background Material

Distributed Systems

- Time/Event ordering
- Synchronization
- Distributed Consensus (Voting)
- Security
  - ▷ Cryptography
  - ▷ Byzantine Generals Problem
  - ▷ Intrusion Detection

Mobile Computing

- Tolerating Disconnection
- Wireless and Ad Hoc Networking
- Power Management
- Security (link layer)

User Interface Design, aka Human Computer Interaction (HCI)

Embedded and Real Time Systems

Introduction to Fault Tolerance

Failures - Cause a machine to give the wrong result for some inputs

- Persistent or Intermittent
- Node Failure vs. Communication Failure
- Security intrusions can be modeled as failures

A formal model of a distributed system

- Modeled as a graph $G = (V, E)$
  - $|V| = N$, i.e. there are N nodes.
  - $|E| \leq \frac{N^2 - N}{2}$, where E is the number of communication channels (links).

A fault tolerant system can continue to operate properly in the presence of a reasonable number of failures.

- Fail Stop - Failed nodes/links shut down
- Byzantine - Failed links/nodes give incorrect values
- Note: undetected faults cannot be tolerated

Fault Tolerance in Distributed Systems

By definition distributed systems don't have a centralized controller

Thus distributed solution methods require reaching consensus (voting)

Distributed systems can be characterized as:.

- Asynchronous - Makes no assumption about timing, no time outs.
- Synchronous - Permits time outs

Fault Tolerance in Asynchronous Systems

Fisher, et al. proved [3] Cannot be guaranteed even under ideal conditions

- Fail stop model.
- Only one failure in N nodes

Why?

- Remember no timing assumptions allowed in Asynchronous Model
- Hence can't time out
- During a long wait for a message or is the node/link just really slow?
- However, G. Bracha and S. Toueg [1] demonstrated that probabilistic consensus is possible
  - ▷ the probability of indefinite delay can be made negligible (have probability 0).

Asynchronous systems are of a more theoretical interest.

- Probabilistic consensus is possible
  - ▷ the probability of indefinite delay can be made negligible (have probability 0).
- Adding failure detectors (so that you know if a node or link is dead) can help.
- Relaxing asynchrony (by allowing atomic operations) helps.

Byzantine Fault Tolerance in Synchronous Systems

Lamport et al. [4] defined the Byzantine Generals Problem (BGP) as:

- Consider a city under siege by N divisions of the Byzantine Army
- Each division has a General.
  - There is one commanding general.
  - The commander has $N-1$ lieutenant generals
- Generals communicate by messengers
- Have to agree on a common strategy (or globally fail)
- What if some generals are traitors? Our goals are:
  - All loyal generals should agree on the same strategy
  - A small number of traitors should not be able to trick the loyal generals into using a bad strategy.

BGP Formalized

One possibility the commander is traitor.

This gives rise to Lamport et al's formalization using Interactive Consistency Conditions

- IC1) All loyal lieutenants obey the same order
- IC2) If the commander is loyal, all loyal lieutenants obey the order he sends.

A question

Consider a case where there is 1 traitor and 3 generals, can we guarantee a correct outcome?

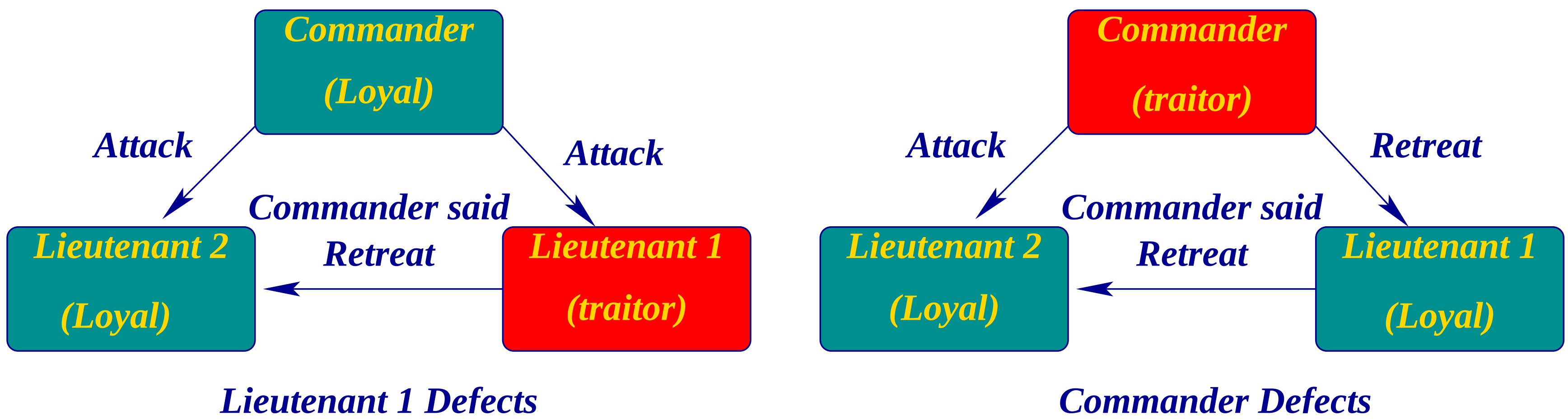
- (HINT) Lieutenants can relay the commander's order.

An Answer

Given: 1 traitor and 3 generals.

To Prove: A correct outcome is not guaranteed

The idea: Prove One Lieutenant Gets Conflicting Reports And Doesn't know what to do



Answer Details

In both cases Loyal Lieutenant 1 receives:

- Attack order directly from Commander
- Retreat order directly from Lieutenant 2

Case 1: Lieutenant 2 defects

- IC2) implies Lieutenant 1 should attack
- Suggests a (faulty) rule: Listen only to the commander

Case 2: Commander defects

- If Lieutenant 1 obeys commander he must attack
- If Lieutenant 2 obeys commander he must retreat
- But this violates IC1)
  - ▷ Thus, lieutenants need to listen to each other to detect a traitorous commander

Generalizing the Result

What if we have $N > 3$ generals and $m < N$ traitors?

To distinguish this from the 3 general Byzantine General Problem we call these generals Albanian Generals.

In general if $N < 3m + 1$, there is no solution

- Suppose $N = 3m$
- Without loss of generality we can model this by partitioning the Albanians
  - 2 Byzantine Lieutenants, each representing m Albanian Lieutenants
  - 1 Byzantine Commander, representing 1 Albanian commander and $m - 1$ Albanian Lieutenants
- But this representation is exactly the unsolvable Byzantine Generals Problem

Approach to Conflicting Messages

So what should a node do if it gets conflicting messages?

Explode in a fiery cataclysm of doom? No...

Each node picks a "representative" message value using a voting method.

- Majority
- Median value
- Mean value (for continuous values)

Picking a voting method depends on application and message type

Approximate Agreement in the BGP 1 of 2

If we have N generals and $m \geq \frac{N}{3}$ approximate agreement is impossible.

Consider a scenario with 3 Generals and one traitor where they

- Have synchronized clocks
- All loyal lieutenants must attack within 10 minutes of each other

This gives rise to modified versions of IC1) and IC2)

- IC1)$'$ All loyal lieutenants must attack within 10 minutes of each other
- IC2)$'$ If the commander is loyal, all loyal lieutenants must attack within 10 minutes of the time given in his order.

Approximate Agreement in the BGP 2 of 2

The commander sends a message with a time

- 1:00 means attack at 1:00
- 2:00 means retreat

Lamport Suggests Each Lieutenant does the following:

- Step 1) If the commander's message is
  - (a) 1:10 or earlier, attack
  - (b) 1:50 or later, retreat
  - (c) Otherwise do step 2
- Step 2) Ask other lieutenant what they decided
  - If the other lieutenant decided, do the same action
  - Otherwise retreat

It can be shown that this approach fails if the commander is a traitor.

Oral Message BGP

Oral messages use a reliable channel where:

- Every sent message is correctly delivered
- The receiver of a message knows who sent it
- The absence of a message can be detected

Lamport et al. developed an Oral Message Algorithm $\mathrm{OM}(m)$, where

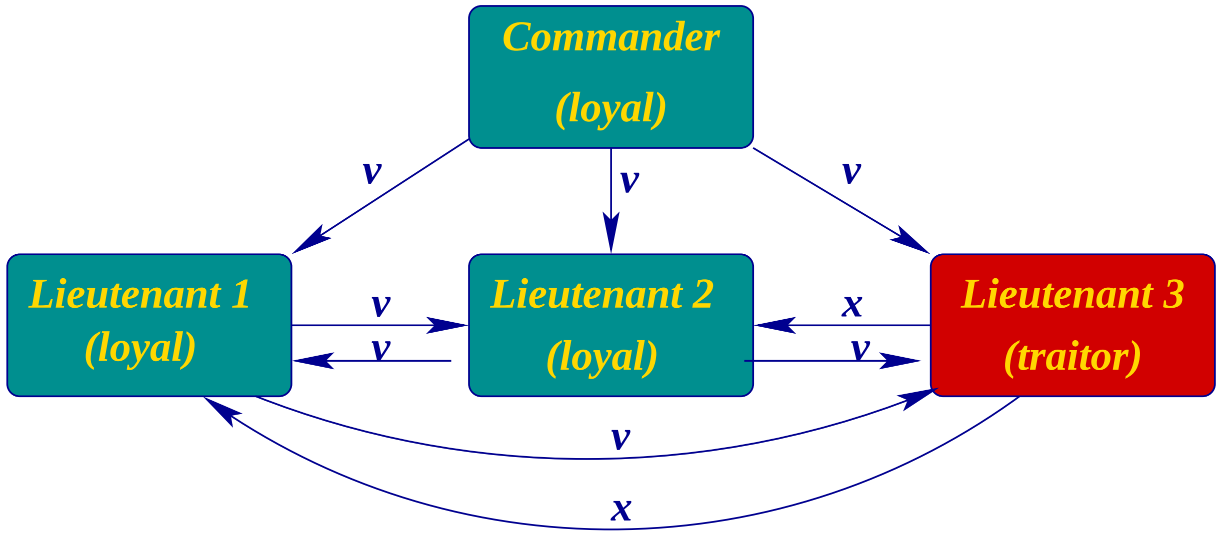
- There are N generals with
  - ▷ 1 Commander
  - ▷ $N - 1$ Lieutenants
  - ▷ m of the generals are loyal
- Each pair of generals has a channel for oral messages
- Can't have too many traitors, requires $N \geq 3m + 1$
- Use a function to obtain representative value $\mathtt{majority}(v_1, v_2, \ldots, v_{N-1})$
  - ▷ Can use simple majority, median for ordered sets or average for continuous values

The Oral Message tolerating m traitors, $\mathrm{OM}(m)$ algorithm

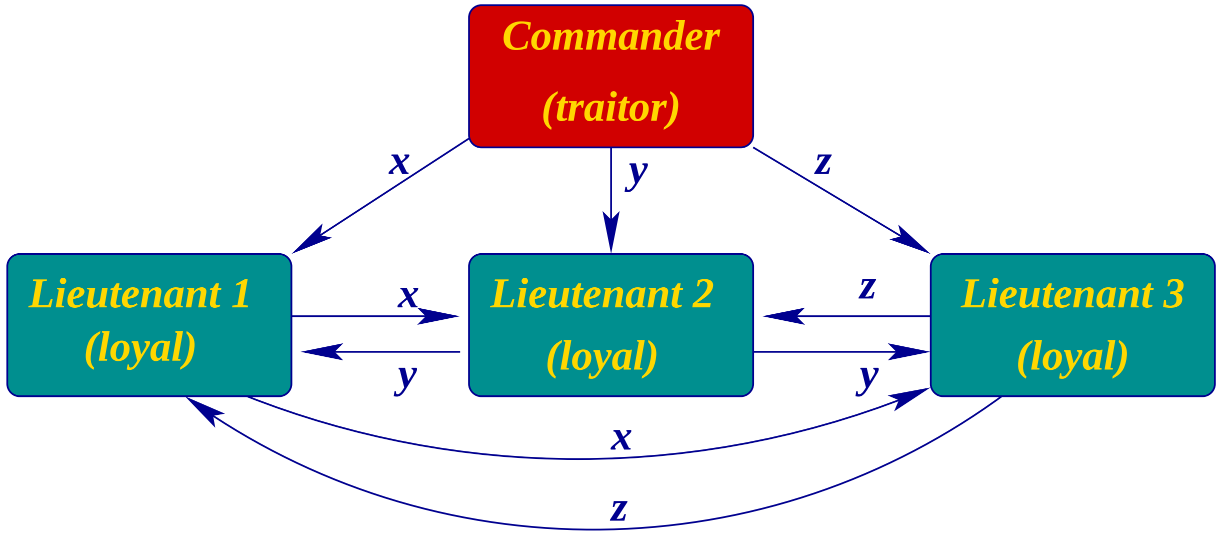
1. $\mathrm{OM}(0)$ ($m = 0$ case, i.e. there are no traitors)
   (a) The commander sends his value to every lieutenant
   (b) Each lieutenant receiving a command uses the value received, if a message does not arrive, uses the value RETREAT

2. $\mathrm{OM}(m)$ ($m > 0$ case, i.e. there are m traitors')
   (a) The commander sends his value to every lieutenant
   (b) For each Lieutenant $i, 1 \leq i \leq N$ let v_i be the value i receives from the commander or RETREAT if no such value was received.
   In the next stage, Lieutenant i will act as a commander of the remaining $n-2$ Lieutenants in $\mathrm{OM}(m-1)$ with order v_i.
   (c) For each node i, let $j \neq i, 1 \leq j \leq N$, be some other Lieutenant. Let v_j be the value j sends to i in Step 2b (using $\mathrm{OM}(m-1)$) or else retreat if he receives no such value.
   Lieutenant i uses $\mathbf{majority}(v_1, v_2, \ldots, v_{N-1})$.

Examples of OM(1) for $N = 4$



Lieutenant 3 Defects



Commander Defects

Remarks on Correctness of $\mathbf{OM}(m)$

Theorem: Algorithm $\mathrm{OM}(m)$ satisfies IC1 and IC2 if there are no more than m traitors and at least $3m$ generals (i.e. $n > 3m$).

Proof by induction on m.

- Base Case: $m = 0$ means there are no traitors, so $\mathrm{OM}(0)$ satisfies IC1 and IC2.
- Induction Step: Show that theorem holds for $\mathrm{OM}(m)$ case if the theorem holds for $\mathrm{OM}(m-1)$ where $m > 0$.
- Case 1: The Commander is loyal.
  - ▷ Lemma: For any m and k, $\mathrm{OM}(0)$ satisfies IC2 if there are at least $2k + m$ generals and no more than k traitors.
  - ▷ If $k = m$ then $\mathrm{OM}(m)$ satisfies IC2 and since the commander is loyal IC1 holds.
- Case 2: The commander is a traitor.
  - ▷ Then there are at most $m - 1$ traitorous lieutenants and 1 traitorous commander.
  - ▷ From our hypothesis are $n-1 > 3m-1$ lieutenants, and $m-1$ traitors. $\mathrm{OM}(m-1)$ on the lieutenants obeys our constraint since $n - 1 > 3m - 1 > 3(m - 1)$.

Some Cost Measures in Distributed/Parallel Algorithms

Common measures of parallel algorithm resource efficiency are:

- Run Time - when the last processor finishes
- Number of rounds (for algorithms that synchronize on iterations).
- Number of messages transmitted
- Operations performed by a single processor
- Work $=$ Operations per processor $\times$ num processors.
- Memory needed (per node or global memory required).

Remarks on Cost/Complexity of OM(m)

- Time: The algorithm runs for $m + 1$ rounds.
  - Work per round is proportional to the number of messages
- Message Count: $O(N^{(m+1)})$.
  - ▷ Round 1: Commander sends $N - 1$ messages
  - ▷ Round 2: $N - 1$ lieutenants act as commanders for $N - 2$ of their peers for a total of $(N - 1)(N - 2)$ messages.
  - ▷ By induction Round $k, 1 \leq k \leq m + 1$ requires

$$\prod_{i=1}^{k}(N - i) = (N - 1)(N - 2)\ldots(N - k) \quad (1)$$

- So the total number of messages is:

$$\text{Number of Messages} = \sum_{i=1}^{m+1}\prod_{j=1}^{i}(N - j) = O(N^{(m+1)}) \quad (2)$$

Concluding Remarks and Alternatives

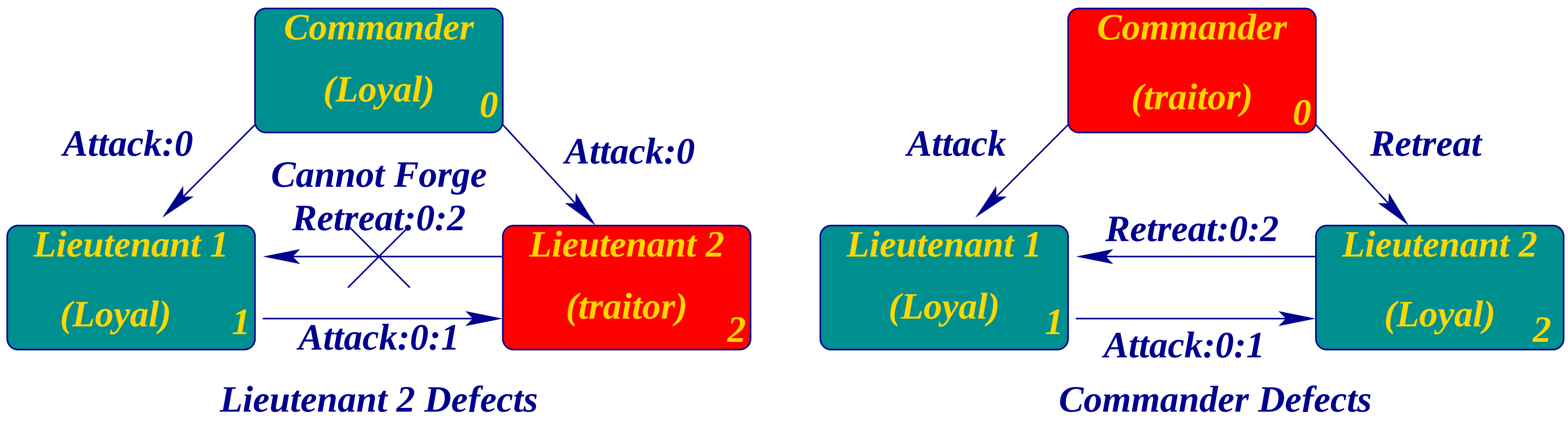
Number of rounds is inherently $m+1$ for this class of problem

- Even if the faults happen to be fail stop instead of Byzantine

Message count is large, since generals must check for altered messages

- If faults are fail stop, the message count can be reduced to (I think to $O(mN^2)$ but I'm not sure).
- Lamport et al [4] developed a written message protocol (assumes Byzantine Faults)
  - ▷ The messages exchanged have tamper resistant signatures appended
    - ▷ Forging signatures is hard (correctly guessing has negligible probability)
    - ▷ Readers of messages can use the signature to detect tampering.
    - ▷ Increases message size
  - ▷ For N generals tolerates up to $m < \frac{N}{3}$ traitors.
  - ▷ Still takes $O(m+1)$ rounds and $O(N^{(m+1)})$ total messages.
  - ▷ Can append signatures to message
  - ▷ In 3 general case, can now detect 1 traitor.
  - ▷ Dolev and Strong [2] were able to reduce the number of messages to $O(N^2)$ messages by avoiding retransmitting messages that were already sent.

Signed Messages Allow Byzantine Agreement with $N = 3m$ Generals



Review and Conclusions

Administrative Details Covered

Brief intro to Ubiquitous Computing (more coverage next lecture)

Review some Background material

- Lampson's paper on your own.
- Byzantine Generals Problem/Distributed Fault Tolerance

Conclusions on Fault Tolerance

- Byzantine Generals Problem is a very strong result
- However, reaching consensus is expensive
- Especially for large systems
- Or systems with expensive data communication
- But some applications need it . . . .

References

[1] G. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. J. ACM, pages 824–840, October 1985.

[2] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. SIAM Journal on Computing, 12:656–666, 1983.

[3] M. J. Fisher, N. A. Lynch, and M. S Paterson. Impossibility of distributed consensus with one faulty process. J. ACM, 32(2):374–382, April 1985.

[4] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. ACM Transactions on Programming Languages and Systems, 4(3):382–401, July 1982. Republished in Advances in Ultra-Dependable Distributed Systems, 1995, N. Suri, C. J. Walter, and M. M. Hugue (Eds.).