

Modeling Concurrent and Probabilistic Systems

Lecture 10: The Alternating Bit Protocol

Joost-Pieter Katoen Thomas Noll

Software Modeling and Verification Group
RWTH Aachen University
noll@cs.rwth-aachen.de

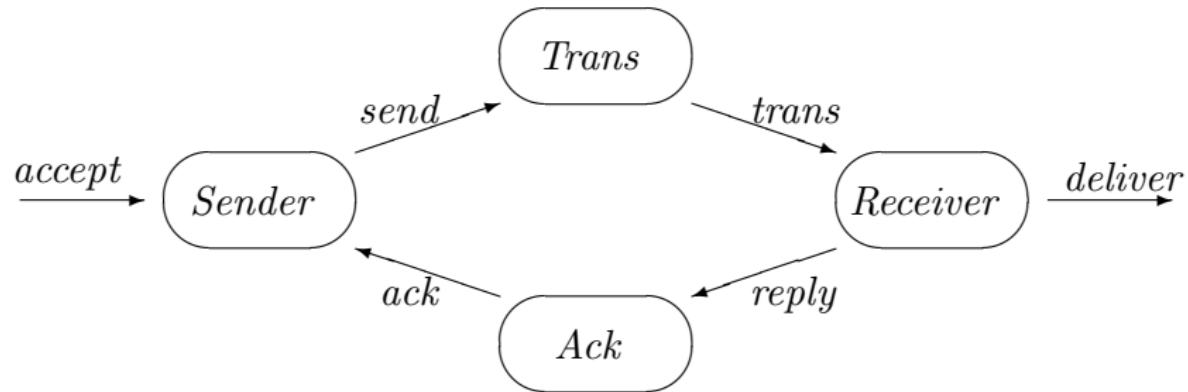
<http://www-i2.informatik.rwth-aachen.de/i2/mcps07/>

Winter Semester 2007/08

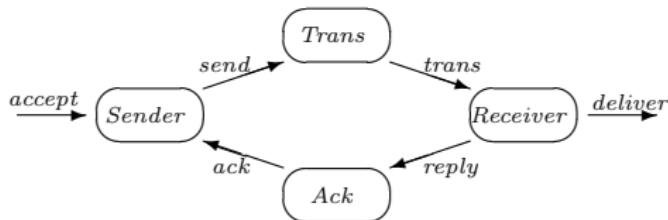
- 1 Repetition: The Alternating Bit Protocol
- 2 Implementation of the Alternating Bit Protocol
- 3 Analysis of the Alternating Bit Protocol
- 4 Extension: Duplication of Messages

Goal: design of a communication protocol which guarantees **reliable data transfer** over **unreliable channels**

Overview of system “architecture”:



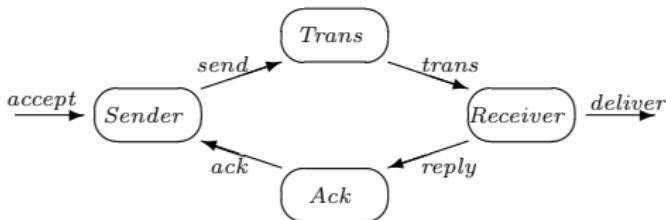
Repetition: Working Principle



- *Sender* transfers data (from a given finite set D) to *Receiver* using channel *Trans*
- *Receiver* confirms reception via *Ack*
- Properties of channels:
 - unidirectional data transfer
 - capacity: one message
(\Rightarrow sequential, i.e., respects order of messages)
 - detection of transmission errors
(loss/duplication/corruption of messages)
 - errors reported to target process

Idea: use redundancy (additional control bit) to ensure safeness of data transfer

Repetition: Modeling of Channels



- *Trans* transmits **frames** of the following form:

$$F := \{db \mid d \in D, b \in \{0, 1\}\} \quad (\text{finite})$$

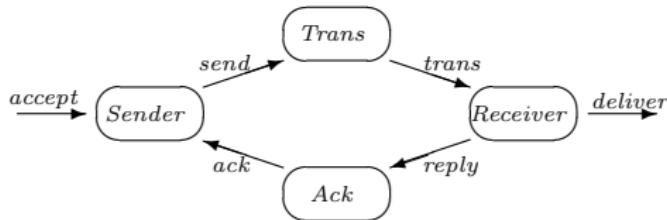
It detects **transmission errors** and reports it to *Receiver*:

$$Trans = \sum_{f \in F} send_f \cdot (\underbrace{trans_f \cdot Trans}_{\text{successful}} + \underbrace{trans_{\perp} \cdot Trans}_{\text{error}})$$

- *Ack* behaves like *Trans* but transmits only **control bits**:

$$Ack = \sum_{b \in \{0, 1\}} reply_b \cdot (\underbrace{ack_b \cdot Ack}_{\text{successful}} + \underbrace{ack_{\perp} \cdot Ack}_{\text{error}})$$

Repetition: Design Goal



Under the above side conditions, give CCS implementations of *Sender* and *Receiver* such that the overall system works correctly, i.e., behaves like a **one-element buffer**:

$$\text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) = \sum_{d \in D} \text{accept}_d. \text{Buffer}_d(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$$

$$\text{Buffer}_d(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) = \overrightarrow{\text{deliver}_d}. \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$$

where

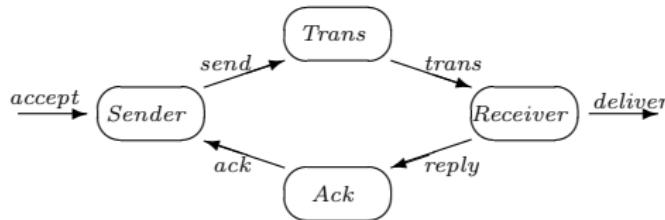
$$\overrightarrow{\text{accept}} := (\text{accept}_{d_1}, \dots, \text{accept}_{d_n})$$

$$\text{and } \overrightarrow{\text{deliver}} := (\text{deliver}_{d_1}, \dots, \text{deliver}_{d_n})$$

$$\text{for } D = \{d_1, \dots, d_n\}$$

- 1 Repetition: The Alternating Bit Protocol
- 2 Implementation of the Alternating Bit Protocol
- 3 Analysis of the Alternating Bit Protocol
- 4 Extension: Duplication of Messages

Implementation of the Sender



Sender accepts $d \in D$ via $accept_d$ and repeatedly sends frames of the form $d0$ over *Trans* until it receives the acknowledgment 0 over *Ack*. For the next data item, control bit 1 is used and so on (\Rightarrow “Alternating Bit Protocol”).

Formally, for $b \in \{0, 1\}$ and $d \in D$:

$$\begin{aligned} \text{Sender} &= \text{Sender}_0 \\ \text{Sender}_b &= \sum_{d \in D} accept_d \cdot \text{Send}_{db} \\ \text{Send}_{db} &= \overline{\text{send}_{db} \cdot \text{Wait}_{db}} \\ \text{Wait}_{db} &= \underbrace{\text{ack}_b \cdot \text{Sender}_{1-b}}_{\text{successful}} + \underbrace{\text{ack}_{1-b} \cdot \text{Send}_{db}}_{\text{error}} + \text{ack}_{\perp} \cdot \text{Send}_{db} \end{aligned}$$

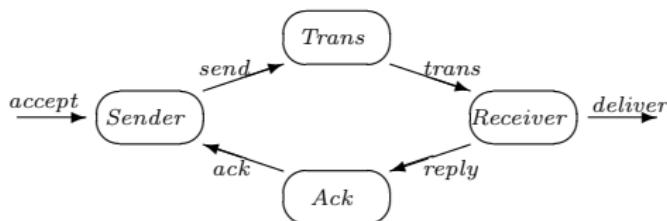
Implementation of the Receiver

Receiver gets frames of the form db or \perp . In the first case, if b has the expected value, d is forwarded via $deliver_d$, and b is returned via Ack . Otherwise the transmission is re-initiated by returning the “wrong” control bit $1 - b$ to *Sender*.

Formally, for $b \in \{0, 1\}$ and $d \in D$:

$$\begin{aligned} Receiver &= Receiver_0 \\ Receiver_b &= \sum_{d \in D} trans_{db}.Reply_{db} \\ &+ \sum_{d \in D} trans_{d(1-b)}. \overline{reply_{1-b}}. Receiver_b \\ &+ trans_{\perp}. \overline{reply_{1-b}}. Receiver_b \\ Reply_{db} &= \overline{deliver_d}. \overline{reply_b}. Receiver_{1-b} \end{aligned}$$

The Overall System



The **overall system** is given by

$$ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) = \text{new } L (\text{Sender} \parallel \text{Trans} \parallel \text{Ack} \parallel \text{Receiver})$$

where

$$L := \{ \text{send}_{db}, \text{trans}_{db}, \text{reply}_b, \text{ack}_b \mid db \in F \} \cup \{ \text{trans}_\perp, \text{ack}_\perp \}$$

- 1 Repetition: The Alternating Bit Protocol
- 2 Implementation of the Alternating Bit Protocol
- 3 Analysis of the Alternating Bit Protocol
- 4 Extension: Duplication of Messages

Theorem 10.1

$$ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \simeq \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$$

Remark: because of internal τ -steps in ABP , $ABP \sim \text{Buffer}$ cannot hold.

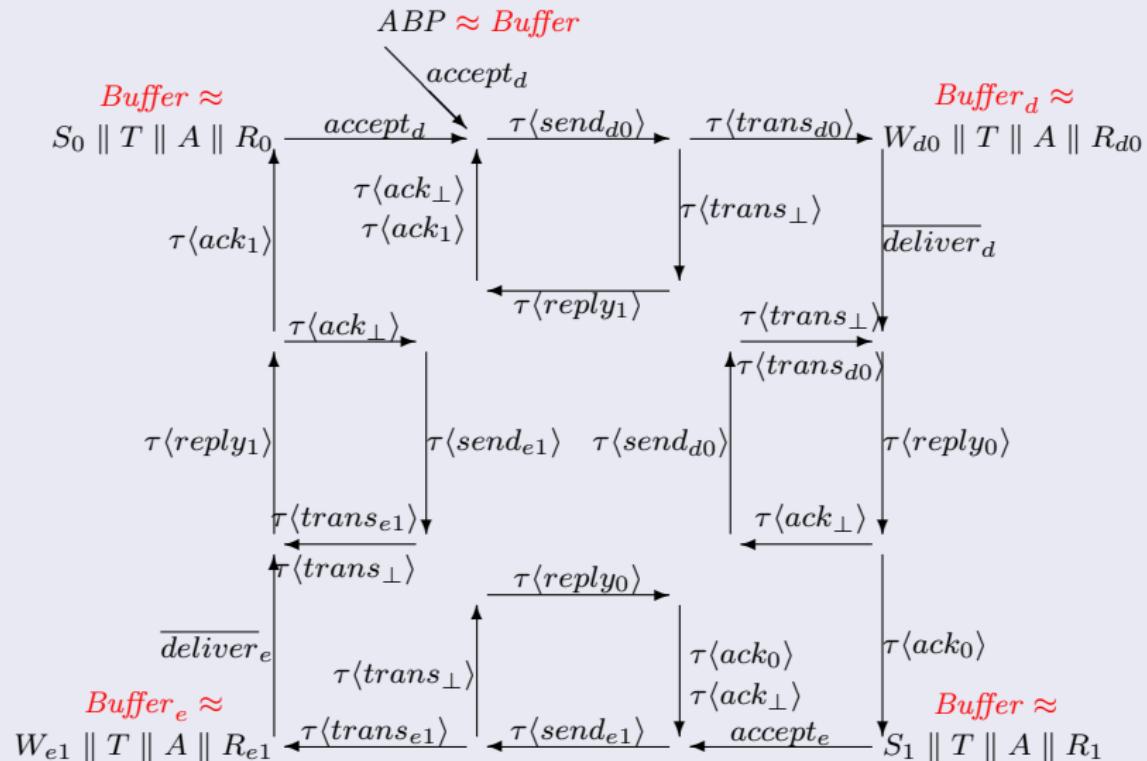
Proof.

- ① Construct transition system of $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
(next slide; $S = \text{Sender}$, $W = \text{Wait}$, $T = \text{Trans}$, $A = \text{Ack}$,
 $R = \text{Receiver/Reply}$, $d, e \in D$; without restrictions)
- ② Show that $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \approx \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
- ③ $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \not\sim \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
 $\implies ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \simeq \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$



Correctness of the ABP II

Proof (continued).



- 1 Repetition: The Alternating Bit Protocol
- 2 Implementation of the Alternating Bit Protocol
- 3 Analysis of the Alternating Bit Protocol
- 4 Extension: Duplication of Messages

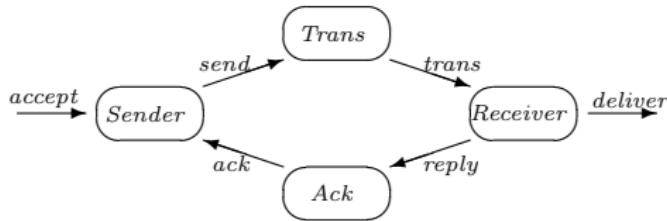
Duplication of Messages I

Duplication of messages can be modelled as follows:

$$Trans = \sum_{f \in F} send_f. (\underbrace{\overline{trans}_f. Trans}_{\text{successful}} + \underbrace{\overline{trans}_\perp. Trans}_{\text{error}} + \underbrace{\overline{trans}_f. \overline{trans}_f. Trans}_{\text{duplication}})$$

$$Ack = \sum_{b \in \{0,1\}} reply_b. (\underbrace{\overline{ack}_b. Ack}_{\text{successful}} + \underbrace{\overline{ack}_\perp. Ack}_{\text{error}} + \underbrace{\overline{ack}_b. \overline{ack}_b. Ack}_{\text{duplication}})$$

Duplication of Messages II



Now the ABP behaves as follows (without restriction):

$$Sender_b = \sum_{d \in D} \text{accept}_d. Send_{db} \quad Send_{db} = \overline{\text{send}_{db}}. Wait_{db} \quad Receiver_b = \sum_{d \in D} \text{trans}_d$$

$$Trans = \sum_{f \in F} \text{send}_f. (\overline{\text{trans}_f}. Trans + \overline{\text{trans}_\perp}. Trans + \overline{\text{trans}_f}. \overline{\text{trans}_f}. Trans) \quad Trans = \sum_{f \in F} \text{send}_f. (\overline{\text{trans}_f}. Trans + \overline{\text{trans}_\perp}. Trans + \overline{\text{trans}_f}. \overline{\text{trans}_f}. Trans) \quad Trans = \sum_{f \in F} \text{send}_f. (\overline{\text{trans}_f}. Trans + \overline{\text{trans}_\perp}. Trans + \overline{\text{trans}_f}. \overline{\text{trans}_f}. Trans)$$

$$Sender_0 \parallel Trans \parallel Ack \parallel Receiver_0 \quad Send_{d0} \parallel Trans \parallel Ack \parallel Receiver_0 \quad Wait_{d0}$$

$$Send_{d0} \parallel Trans \parallel Ack \parallel Receiver_0 \quad Wait_{d0} \parallel (\dots + \overline{\text{trans}_{d0}})$$

Deadlock \implies ABP cannot handle this